PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/14	A2	(11) International Publication Number: WO 99/26153
		(43) International Publication Date: 27 May 1999 (27.05.99)

(21) International Application Number: PCT/US98/24741

(22) International Filing Date: 18 November 1998 (18.11.98)

(30) Priority Data:
60/065,664 18 November 1997 (18.11.97) US

(71) Applicant: STELLAR COMPUTING [US/US]; Suite 301, 1270 Oakmead Parkway, Sunnyvale, CA 94086 (US).

(72) Inventor: ZHU, Min; 24920 La Loma Court, Los Altos Hills, CA 94022 (US).

(74) Agents: HAMRICK, Claude, A., S. et al.; Oppenheimer Wolff & Donnelly LLP, Suite 600, 10 Almaden Boulevard, San Jose, CA 95113 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD FOR ESTABLISHING A COMMUNICATION CONNECTION BETWEEN TWO OR MORE USERS VIA A NETWORK OF INTERCONNECTED COMPUTERS

(57) Abstract

A method for establishing a communication link between two or more users via the Internet, and, more specifically, a web-server based real-time data conferencing system, is disclosed. Under the presently preferred embodiment, a user uses an application (such as a web browser) to retrieve and view a web page. On the web page (or the like), a clickable icon or hyperlink is provided to call another user or service. At this time, there is no custom software on the user/caller side. However, the hyperlink may activate a subprogram to gather certain types of information of interest to pass to the server to tailor a response; it may also perform any other tasks as necessary. The server (call-center software), upon receiving the request to establish a connection, determines an agent to receive the call and activate any related software thereof. In the presently preferred embodiment, the call center processes the request and activates an application on the agent's machine to establish the connection. More specifically, it causes the agent's web browser to be launched if it is not already launched and it provides a customized web page to the caller if appropriate. In the next step, now having established a connection between the caller and an agent, a collaboration software can be activated to allow better communication between the two parties. This collaboration software can be a chat program, a white board program, Internet phone, or any other communication software. Under this paradigm, the caller does not need any software to establish a connection with an agent who has the necessary software for receiving and processing the call.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD FOR ESTABLISHING A COMMUNICATION CONNECTION BETWEEN TWO OR MORE USERS VIA A NETWORK OF INTERCONNECTED COMPUTERS

FIELD OF INVENTION

The present invention relates generally to methods for establishing a communication channel between two or more users, and, in particular, call center methods for establishing a real-time communication channel between one or more users to a central processing center.

BACKGROUND

Under the concepts and software applications of the prior art, two users wishing to communicate over the internet (or any other communication link) would have to arrange a meeting time and an agreed upon mode of communication, or that one party would have to always wait for the other party to call. These models are inefficient in that time has to be spent in arranging the meetings and that they do not allow spontaneous calling of another user over the communication links.

In another real model, the call center model, customers call a typically toll free number for service or inquiry on a given product or service. For example, a customer may call a toll-free number for an insurance company to inquire about an insurance quote, and the customer will then be routed to an available service representative. If no representative is available, the customer will be placed on hold for the next available representative. Once a representative becomes available, the customer is connected to the representative. This real life model is an efficient model based on the telephone network. However, with the advent of the internet, the telephone network has become an out-of-date model.

It would be desirable to have an internet based call center model for handling calls from a number of customers "calling" from their computer terminals or a telephone equivalent call from one user to another user through the internet.

SUMMARY OF THE INVENTION

Briefly, the presently preferred embodiment of the present invention provides a method for establishing a communication link between two or more users via the internet, and, more specifically, a web-server based real-time data conferencing system. Under the presently preferred embodiment, a user uses an application (such as a web browser) to retrieve and view a web page. On the web page (or the like), a clickable icon or hyperlink is provided to call another user or service. At this time, there is no custom software on the user/caller side. However, the hyperlink may activate a subprogram to gather certain types of information of interest to pass to the server to tailor a response; it may also perform any other tasks as necessary. The server (call-center software), upon receiving the request to establish a connection, determines an agent to receive the call and activate any related software thereof. In the presently preferred embodiment, the call center processes the request and activates an application on the agent's machine to establish the connection. More specifically, it causes the agent's web browser to be launched if it is not already launched and it provides a customized web

page to the caller if appropriate. In the next step, now having established a connection between the caller and an agent, a collaboration software can be activated to allow better communication between the two parties. This collaboration software can be a chat program, a white board program, internet phone, or any other communication software.

Under this paradigm, the caller does not need any software to establish a connection with an agent who has the necessary software for receiving and processing the call. This paradigm is particular helpful in service and/or sales types of activities where a customer inquiry can be answered in a chat session over the internet without requiring the customer to have any software to begin with.

The caller can also download the call-center supported software on its machine. As long as the software is running at a connection that can be determined and be reached by the call-center software, anyone can request a connection with this particular caller and be connected to this particular caller. Thus, this model is very much like the ubiquitous telephone at every household. As long as the phone is connected and not be off-hooked, another party can call in and be connected when answered. Even if one party is not available at a particular time, the calling party can be recorded and be notified to the receiving party when the receiving party comes on-line.

These and other features and advantages of the present invention will become well understood upon examining the figures and reading the following detailed description of the invention.

DRAWINGS

Fig. 1 illustrates a diagram showing the collaborating systems and the various components thereof.

Fig. 2 illustrates a block diagram of a web server interface of the preferred embodiment.

Figs. 3a and 3b illustrates examples of web server interfaces.

Fig. 4 illustrates a server architecture of an embodiment.

Fig. 5 illustrates a block diagram of data collaboration services.

Fig. 6 illustrates a block diagram of a session manager.

Fig. 7 illustrates a block diagram of an intelligent routing server.

Fig. 8 illustrates a block diagram of a web client manager.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to Fig. 1, a block diagram divided into three sections, a visitor section 10, a server section 12, and a member section (or registered section) 14 is provided to illustrates the three collaborating systems and the various components provided therein. The visitor section 10 can be considered as the computer system being used by a guest (or visitor or customer or user). In the visitor section 10, a guest/user activates an application of the preferred embodiment through a regular web page 16 or activates a button to activate the application via a browser 18. Note that the browser here can be a commercially available browsers or it can be custom browser provided specifically for the current task. At the visitor section 10, there can be also a suite of collaboration software applications such as text exchange (chat) program, a whiteboard program, a web page tour program, a web page push program, or an application sharing program. These program or programs can be downloaded real-time from a server upon request.

In the server section 12, there may be a customer web server 20 for handling regular web page access. There can also be a dedicated server 22 for the handling the processing steps of the presently preferred embodiment. In this dedicated server 22, there can be a web object request broker for handling the various types of requests. For a call center request, the request is routed to a call center 26 which provides administration of the in-coming calls, for managing the various sessions between a guest/user and a call center agent (registered member), and for tracking accounting related issues such as time and billing. For center types of requests, the call center 26 provides data from a context repository 28 which in turn provides dynamic or static data associated with the request, which is associated with a particular hyperlink. In other words, customized data may be provided to the guest depending on the particular request. The call center 26 interacts with an agent server 30 which manages a number of agents, each performing a pre-programmed task such as configuration verification, virus check, etc. There is also a communication server 34, interacting with a collaboration server 32, for connecting multiple users through the collaboration applications indicated at 31 and 40.

For the member section 14, which are the computer systems for the users being connected to, there may be a registration component 36 for registering guests/users. At this section, there may be one or more call center agents (registered members) for connecting by the call center 26 to a guest/user. Generally speaking, these call center agents are software providing an interface to allow a person to communicate to the guest/user via the collaborative applications 40.

Fig. 2 illustrates the interface scheme of the web object request broker 24 of Fig. 1. Here, the server dispatcher 50 connects to the call center (Fig. 1, 26) on one side and connects to an object request broker on the other side 52. The object request broker 52 communicates to the object request broker 54 of a web server interface 56 for communication with web servers 58. The web server may any one of the commercially available servers from a number of sources. In essence, this figure illustrates the manner for the call center (Fig. 1, 26) to communicate with the web servers.

Figs. 3a and 3b illustrates two examples of different web server interfaces of Fig. 2. Fig. 3a illustrates the use of an NS Enterprise Server 60 interfacing with an object request broker (ORB)-based web application interface 62 to the server dispatcher 64 (Fig. 2, 50). Fig. 3b illustrates an Oracle Web Server 66 interfacing with an ORB-based cartridge interface 68 to the server dispatcher 70 (Fig. 2, 50).

In operation, certain buttons on a web page are provided and associated with the application software of the preferred embodiment. The web page may be part of a secured web site requiring a password for access. The activation process may also be provided through the activation of an application on the user's machine. Upon activation, a communication link to the internet is provided to carry out the preferred method of the present invention. More specifically, the server of the preferred method of the present invention (server section of Fig. 1) takes over and sends a message to one or more registered members (or call center agents) on the receiving side (member section of Fig. 1) to signal that there is a guestj/user waiting to be connected. Depending on the mode of operation, a registered member may decide to answer the signal or refuse the signal. In this manner, this is a model like a telephone which the receiving person may decide to answer the call or refuse the call. In a call center model, the registered member may not have a choice and may have to answer the call. At the time of the connection, the system may offer the choices for a phone connection (through internet phone or regular phone) or provide information from a database regarding the user/guest. A live data connection between the user/guest and the registered member may be set up through the use of collaboration

software such as a chat program, a whiteboard, web page navigation, or perhaps push technology programs. At this time, there is a real-time conference. Optionally, the system may allow other guests/users or registered members to join this conference. Note that call center agents or registered members are users that have registered with the system as being available for connection to other guests/users or registered members and therefore may be "called" by other people. A guest/user without application software of the preferred embodiment can not be called by others and may only call registered members. The above described method provides the capability for real-time connection to a database where information regarding a guest/user may be readily retrieved and shown to the registered member prior to connecting to the guest/user. Additionally, a guest/user may request a specific registered member for connection rather than being randomly assigned.

Other features of the preferred embodiment include readily connection to the telephone network where the guest/user may be directly connected to the registered member for a live conversation. This may be achieved through the use of a second modem on the user side or the registered members side for direct dialling to the other side. Further note that a user in one embodiment need not to have any specialized software on its machine. In an alternative embodiment, the guest/user may download an agent for receiving and identifying incoming or outgoing calls.

With the use of a database, a number of transactions can be recorded and automated. For example, a registered member may set a reminder in the database for reminding the registered member with regard to a particular transaction over an particular item of interest with a particular user/guest. This reminder may be set in the transaction database or the database for the item of interest. In this manner, potential sales leads can be tracked and followed up.

The above described preferred embodiment and method can be further customized to tailor to specific applications. For example, in one application (active meeting), the application can be tailored to be a specific conferencing application where guests and/or registered members select a designated registered member as the agreed meeting place. A real-time interactive chat session can be provided such that everyone can interact. In another application (active connections), the present invention can be tailored to be a commerce application where guests/users visiting can be directly connected to a randomly-selected, a user-requested, or a best-fit service representative (registered member). In the best-fit situation, the guest/user may have answered to a number of questions generally describing the request or problem. Based on this information, an intelligent database subsystem can examine the answers and determine the most qualified representative for this particular guest/user. In the user-requested service representative situation, a list of service representative is provided to the guest/user. Upon selecting a particular service representative, the guest/user is connected to the particular service representative via the TCP/IP address of the service representative. In this application, if appropriate and available, user information, can be readily retrieved from the database and provided to the registered member as background information. Furthermore, guest/user may be allowed access to a number of databases, including a scheduling database for scheduling with a particular service representative (registered member) or to retrieve and enter information into a database. In yet another application, a registered member can call up a guest/user, through the use of an agent at the user/guest side, and interact with the guest/user.

Fig. 4 illustrates one server architecture implementation for the preferred embodiment. Here, the interface to the web is through a web server 80. The web server 80 interacts with a number of components of the preferred embodiment, including an active meeting component 82, any custom applications 84, or OEM

applications 86. A session manger and web client manager 88 interfaces and interacts on one side with active meeting, custom applications, and OEM applications, and interfaces and interacts with an intelligent routing component 90, a managed messaging component 92, a resource scheduling component 94, and an enterprise integration component 96. All of these components interacts with real-time collaboration services 98 and telephony services 100 which rest on an operation system and database 102.

Referring to Fig. 5, further describing data collaboration services of the preferred embodiment, internet protocol based data and servers 110 interfaces with a multi-point communication service 112 for interfacing a number of guest/users and registered members. Conference control 114 is put in place to properly route and control access to various services, including chat programs, presentation programs, document review programs, web tour programs, interactive forms programs, viewer programs, application sharing programs, netmeeting program, desktop sharing programs, etc., all of which, in one aspect, may access data from the data collaboration server 116 and in another aspect access CTI servers.

Fig. 6 illustrates the session manager, which can be part of the call center or agent server of Fig. 1. Here, the session manager interfaces with a master collaboration clustering server 122 and manages a billing and accounting module 124 and database 125, a directory service module 126 which interacts with a database 121 and an active directory service 127, an object storage module 128 working with an enterprise data depository 129, a security module 130 providing security and interfacing with public key infrastructure 131, and an archive module 132 working with a database 133 and an active directory service 134.

In routing the guests/users, referring to Fig. 7, an intelligent routing server 140 manages call queues 142, a call handler 148, and third party ACD 154. The call handler 148 interacts with community servers 144 and distributed ACD 146. The intelligent routing server 140 also provides managed messaging service 156.

Fig. 8 illustrates a web client manager model where a web client manager 160 manages a multi-point data manager 162 (described above), automatic download and version control subsystem 164, a session manager 166, a token manager 168, and a phone manager 170. The multi-point data manager 162 manages the data cache for each session 172 and 174. The automatic download and version control subsystem control the agents residing on the computers of guests/users and registered users. The session manager 166 manages the sessions. The phone manager 170 manages the phone connections.

Appendix A further provides technical detail with respect to the embodiments of the present invention. Appendix B further describes the features of the preferred embodiments. Both appendices are part of the embodiments of the present invention.

Although the present invention has been described in terms of specific embodiments it is anticipated that alterations and modifications thereof will no doubt become apparent to those skilled in the art. It is therefore intended that the following claims be interpreted as covering all such alterations and modifications as fall within the true spirit and scope of the invention.

I/We claim:

APPENDIX A

Call Center Design

Terms and definition:

Member
: The guy who have registered in Call Center, have a account and can access resource on Call Center

Visitor
: User who click the button on the web page that our button creator created.

Call Center Server
: The software server make member and visitor negotiate together.

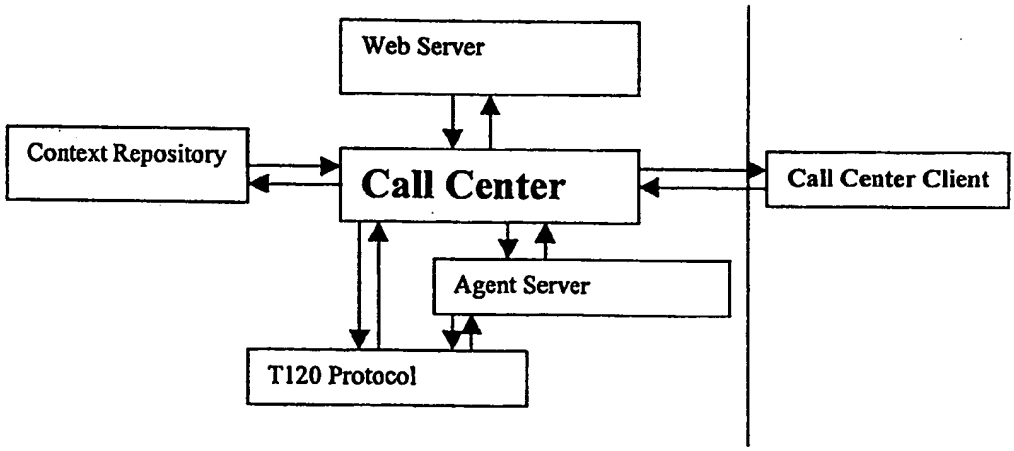
Call Center Client
: The response layer to Call Center in client site.

Web Server Interface
: Interface that web server and Call Center cooperator together.

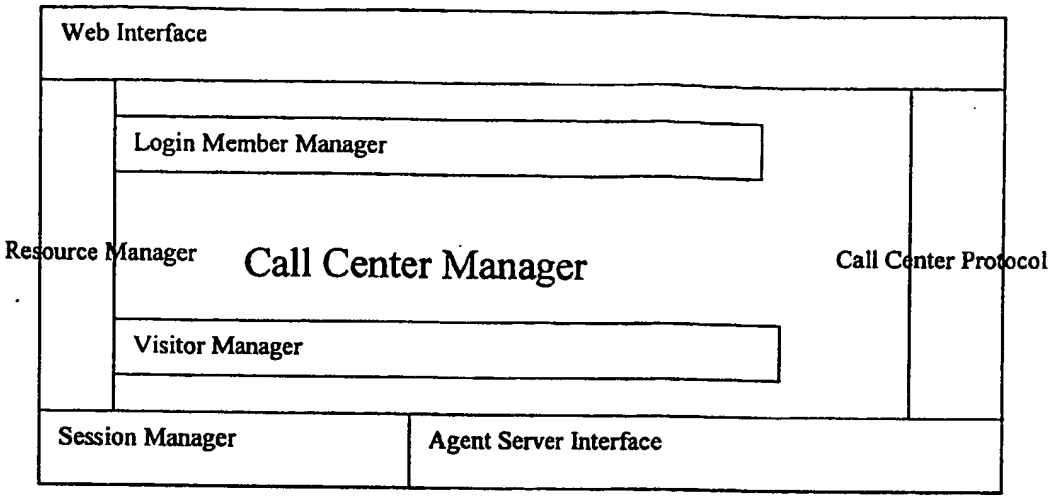
Resource Manager
: Use a DB server as back support, provide several member, visitor, page and other resource service to Call Center.

Session Manager
: Manage the session like ARM of T120.

CC Environment Diagram:



CC Internal Structure Diagram:



Requirement between objects

WEB INTERFACE

Through Web Interface.

Web Interface will interpret string send from web, translate them to data struct recongnize by call Center Manager, and call associate server rutine provide by call center manager.

Web Interface will reponse the requirement of Call Center Manager to dynamically create page send to the web server, The web server then will send this HTML or ASP page to user as a result.

So the function will be implement in Web Interface is:

1.Translate string stream sand from web server to a C or C++ object used by Call Center Manager.

2.Dynamicly create HTML or ASP page to web server.

For detail, see Chapter "Format of string stream send from web server"

Chapter "Reply Page templates"

Chapter "Data structure used in Web Interface"

Resource Manager

Resource Manager will save and provide the following information:

1.Member Information.

2.Accounting Information.

3.Statistic Information.

4.Page Information.

5.Field Information.

6.Visitor Information.

Resource Manager will provide service to call center to:

1.Verify a validate user.

2.Check the member's accounting, status

3.Check the member's priority, which resource(page, member...) can be accessed.

4.According to a simple query, return a result to call center.

5.Save visitor information for to be queried by member.

For Detail, See:

Chapter: Resource Manager

Member Profile format.

Accounting solutions.

Statistic Information format.

Page Information format.

Field Information format.

Visitor Information format.

Agent Protocol:

Provide a private protocol between Call Center and the Call Center Client.

Use the senddata, OnDataIndication service provide by Agent Server, connect with call center client, Let member to see how many visitor is online, modify the profile, give administrator message, run session. terminal session, Set current status. Query current accounting, and download new session app. Logout.

Agent Server Interface:

Provide a simple connect function. Under Construction....

Session Manager:

Provide session service, and manage session.
The service provide to Call Center:
CreateSession
LeaveSession
TerminateSession.
QuerySession

CallCenter Client:

Under construction;

Chapter 1: Web Interface.

Section 1: Format of String Stream Send From Web Server

The string stream send from web server is the following styler:

URL + parameter1 + parameter2 +...

A sample:

http://www.stellar.com/www/tom0.asp, T= R, user=tom , address=202.47.133.196, password=abcde

we assume that the first parameter is the message type, there are the following message type:
T=R, this is a register message.
T=V, this is a visitor message.
T=L, this is a log in message.
T=P, this is a pay message.
For T=R, this is a register message.
There are the following Parameters
address, required
city, required
state, required
zip code, required
company,
department,
first name, required
middle name,
last name , required
title
email_address0, required
email_address1
phone_number 0, required
phone number 1,
fax number,
[page id0
[[fld_fldname
operator
fld value]...]...]

For T=V, that is a visitor message.
Parameter is the following:
Page: the URL of this page.
Address: the user TCP/IP address

filed name=field value,
[field name=field value, ...]

For T=L, it is a member log in message.
ID is the user id create by resource manager,
PASSWORD is the password given when register, compare with the password provide by resource manager

For T=P, it is a expire member want to pay money to by time.
ID is the user id.
PASSWORD is the password.
Create card # or phone bill, under construction...

Section 2: Reply Page Templates

There are 10 types of page template used to dynamically create home page to reply web server.

1.At lease one required field is empty.

2.Pay money by credit card or by phone page, ...(under construction)

3.Registed successfully and give user a password, a user id.Let user down load the client part of stellar server, give user a licence agreement.

4.Login successful page, will or will not show statistic information.

5.Expired page, when a user use out of his/her time resource, and try to log-in in 30 days, this page will be given to user indicate that user is run out of time, in this page, user can pay for the time.

6.Unauthorized user, this situation may be by user mis-typing the password, can give user a chance to retry or registry a new member.

7.Every body is busy page, this means there is no member is idle, so this page will say sorry to user and let him retry.

8.wrong create card# wrong page, this means the user mis-type the credit card number. Will ask user retype the register page.

9.Chat page.parameters is under construction....
Parameters:
1.INT userId, indicate the user id
2.CHAR* server TCP/IP address.

10.white board page.parameters is under construction....
Parameters:
1. INT userId, indicate the user id.
2. CHAR* server TCP/IP address.

Section 3: Data structure Used by Web Interface

```
//Operator use in filter
typedef enum
{
        CCOPT_GT,               //for string, number, float
        CCOPT_EQ,               //for string, number, float
        CCOPT_LT,               //for string, number, float
        CCOPT_GET,              //for string, number, float
        CCOPT_LET,              //for string, number, float
        CCOPT_CONTAINS          //for string only.
}CCOperator;

//access field define item
typedef struct tagCCAccessFieldDefinitionItem
{
        UINT field;
        CCOperator operation;
        BYTE value[MAX_FIELD_VALUE_LEN];
}CCAccessFieldDefinitionItem;
 //access resource
typedef struct tagCCAccessableResourceItem
```

```
{
        BYTE url[MAX_URL_LEN];
        UINT count;
        CCAccessFieldDefinitionItem* access_field_definition_list;
}CCAccessableResourceItem;

//Access resource table
typedef struct tagCCAccessableResourceList
{
        UINT count;
        CCAccessableResourceItem* accessable_resource_list;
}CCAccessableResourceList;


typedef struct tagCCMemberInfo
{
        ULONG id;
        BYTE password[MAX_PASSWORD_LEN];

        //company address
        BYTE address[MAX_ADDRESS_LEN];
        BYTE city[MAX_CITY_LEN];
        BYTE state[MAX_STATE_LEN];
        BYTE postal_code[MAX_POSTAL_CODE_LEN]
        BYTE company_name[MAX_COMPANY_NAME_LEN];
        BYTE department_name[MAX_DEPARTMENT_NAME_LEN];

        //person name and title
        BYTE first_name[MAX_MIDDLE_NAME_LEN];
        BYTE middle_name[MAX_LAST_NAME_LEN];
        BYTE last_name[MAX_FIRST_NAME_LEN];
        BYTE   title[MAX_TITLE_LEN];

        //person address, include email address, phone number ect.
        BYTE email_address0[MAX_EMAIL_ADDRESS_LEN];
        BYTE email_address1[MAX_EMAIL_ADDRESS_LEN];
        BYTE phone_number0[MAX_PHONE_NUMBER_LEN];
        BYTE phone_number1[MAX_PHONE_NUMBER_LEN];
        BYTE fax_number[MAX_PHONE_NUMBER_LEN];

        //default app session would automatic lanuch up.
        CCSessionType default_lanuchup_session;

        //access right..., special which URL the listen is interest in, and
        //which field the listen is interest in, and the value scope.
        //..
        CCAccessableResourceList accessable_resource_list;

        //current status.
        CCListenerStatus status;

        //status when run.
        CCListenerRunTimeStatus run_status;
}CCMemberInfo;
```

Chapter 2: Resource Manager

Section 1: Member Profile Format.

See CCMemberInfo.

Section 2: Accounting information.

Format:

Field Name	Field Type	Field Len(for string)
UserID	ULONG	
Login time	datetime	
Log out time	datetime	

Field Name	Field Type	Field Len(for string)
UserID	ULONG	
PayedMoney	$	
PayedDate	datetime	
PayedType	int	

When Call Center Send Accounting Information?

Call Center will Call SaveAccountingInfo when a user log out, or Call Center find that user is out of time resource.

Accounting information will be quire when user want or call center it self check the resource, User will check how many times remain, how much money remain in account, user will ask for a time usage table.

Interfaces provide to Call Center.

CCError SaveAccountingInfo(INT userId, ULONG loginTime, ULONG logoutTime);
CCError GetRemainTime(INT userId, ULONG* timeRemain);

Section 3: Statistic Information Formation

Under construction.

Section 4: Page and Field Information Format.

Format:

Fld Name	Fld Type	Fld Len(for string)
Id	number	
URL	string	MAX_URL_LEN

Fld Name	Fld Type	Fld Len(for string)
Id	number	
FldName	string	MAX_FIELD_NAME_LEN
FldType	number	

When Call Center use this information?

When a vistor click a button, the button will give server server information, Call Center will get this information through ORB or not, then first, Call center will check if the page is registed by call resource manager function: HasThisUrl, if Resource Manager report that it is a valid URL, get it ID, and try to find a listener, will call IsThereMemberListenThisPage that time, if not, return a page indicate user wait an retry, If Find,

Section 5: Visitor Information Format.

Under Contruction.

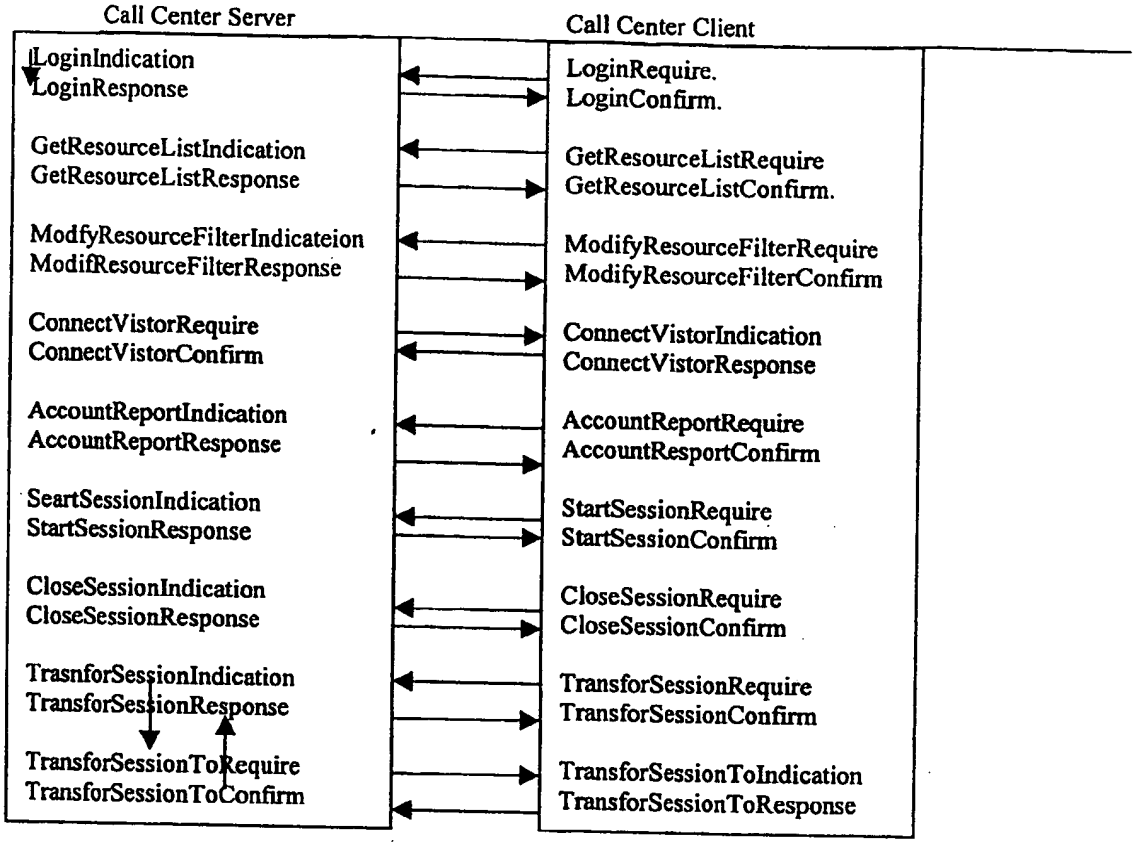
Chapter 3: Call Center protocol

0.Issues

When the call center client launch? launch when windows/winnt startup?I assume the call center client has launch up and waitting the call center server awake it.

1.primitives

After register a sap for this member, Call Center will create a sap on agent server, and through this sap, Call Center server will negotiate with the Call Center Client. If Call Center Client Send a Log out packet, Call Center Server will unregister this sap.



Chapter 4:Session Manager

1.primitives.

CreateSession
CloseSession
GetSession
TrasnferSession

Chapter 5: Call Center Object
Call center

Chapter 6: Call Center Client

Call Center Client is the client side program that is in the layer with Call Center Server,

Chapter 5:Call Center Object

Call Center object will be given formatted data from the web interface, query and retrieve data from resource manager, setup connection with Call Center Client, and manager session. Response member's Require.

```
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*  Stellar server stack     Project                                       */
/*  WEB server interface implement file                                    */
/*                                                                         */
/*  WEBIO.CPP                                                              */
/*                                                                         */
/*  Copyright (c) 1997 Stellar Computing Corp.                             */
/*  All rights reserved                                                    */
/*                                                                         */
/*-------------------------------------------------------------------------*/
////////////////////////////////////////////////////////////////////////////
//Create Date:  06/24/97
//Code Creater: Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//

#include "WEBIO.H"
#include "web.h"

/*
In web.h, there are 2 prototype:
typedef void (*funcWebCallBack)(char* webData, ULONG userDefine);
WEBRegisterCallBack(funcWebCall Back, ULONG userDefine);
*/

////////////////////////////////////////////////////////////////////////////
//Purpose:Web call it to notify the web interface some data need to precess.
//
//Parameters:
//  webData:    webData string, must terminate by a zero char.
//  userDefine: define by user, web will simplely copy this data to this para-
//              meter valued when register.
//
//Create Date:  06/24/97
//Code Creater: Tom.Wang
//Modify Record:
//      Modifier:
//      Reason:
//      Date:
//      Detail:
//
void WebCallBack(LPCSTR webData, ULONG userDefine)
{
    ASSERT(IsGoodReadPtr(webData, 1));
    ASSERT(IsGoodReadPtr(userDefine, sizeof(CWebInterface)));

    ((CWebInterface*)userDefine)->InterpretWebStream(webData);
}

////////////////////////////////////////////////////////////////////////////
//Create Date:  06/24/97
//Code Creater: Tom.Wang
//Purpose:Construct a web interface object.
//
//Parameters:
//  func:   the func given by call center, to get formated data from web interface
//
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CWebInterface::CWebInterface(funcCCCallBack func)
{
    ASSERT(IsGoodCodePtr(func));

    m_CCCallBack = func;

    WEBRegisterCallBack(WebCallBack, this);
}

////////////////////////////////////////////////////////////////////////////
//Purpose:
//  Interpret web stream, format the stream and call call back func to give
//  the formated data to call center.
//
//Parameters:
//  webData: a data send from web server, the detail mean see callcent.doc,
//      chapter: web server interface.
//
//Create Date:  06/24/97
//Code Creater: Tom.Wang
//Modify Record:
//      Modifier:
```

A-11

```
//      Date:
//      Reason:
//      Detail:
//
void CWebInterface::InterpretWebStream(LPCSTR webData)
{
    CCWEBRequireType reqType;
    CHAR* pData = new CHAR[30000];//should modify.

    //<TBD>

    (*m_CCCallBack)(reqType, pData);
}
```

A-12

17

```
/*-------------------------------------------  ----------------------*/
/*                                                                   */
/*  Stellar server stack    Project                                  */
/*  WEB SERVER INTERFACE WITH CALL CENTER SERVER                     */
/*                                                                   */
/*  WEBIO.H                                                          */
/*                                                                   */
/*  Copyright (c) 1997 Stellar Computing Corp.                       */
/*  All rights reserved                                              */
/*                                                                   */
/*-------------------------------------------------------------------*/
////////////////////////////////////////////////////////////////////////////
//Create Date:  06/24/97
//Code Creater: Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//
#ifndef   WEBIO_H
#define __WEBIO_H_

class CWebInterface
{
public:
    CWebInterface(funcCCCallBack func);
    ~CWebInterface();
/*
<TBD>
*/

    void InterpretWebStream(LPCSTR webData);
    UpsendPage(CCUpsendPageType type, BYTE* data);
//....
private:
    funcCCCallBack m_CCCallBack;
};/*CWebInterface*/

#endif/*__WEBIO_H_*/
```

A-13

```
/*-------------------------------------------  ---------------------------*/
/*                                                                        */
/*  Stellar Server Stack    Project                                       */
/*  resource management head file                                         */
/*                                                                        */
/*  RESMGR.H                                                              */
/*                                                                        */
/*  Copyright (c) 1997 Stellar Computing Corp.                            */
/*  All rights reserved                                                   */
/*                                                                        */
/*------------------------------------------------------------------------*/
////////////////////////////////////////////////////////////////////////////
//Create Date:  06/18/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//
#ifndef   RESMGR_H
#define __RESMGR_H_

////////////////////////////////////////////////////////////////////////////
//context depository interface between call center.
//
class CResourceManager
{
public:
    CResourceManager();
    ~CResourceManager();
//accounting.
    CCError SaveAccountingTimeInfo(CCMemberID memberId,  CCTime loginTime,
        CCTime logoutTime);
    CCError GetRemainTime(CCMemberID memberId, ULONG* timeRemain);
    CCError WriteAccountingReport(INT memberId, LPCSTR fileName);

//member.
    CCError RegisterMember(CCMemberInfo* pMemberInfo);
    CCError UpdateMemberInfo(CCMemberInfo* pMemberInfo);
    CCError IsThisVistorMatchAnyMember(CCVistor* pVistor, INT* pCount, CCMemberID* pId);
    CCError MemberLogin(CCLogin* pData);
    CCError GetMemberDetailInfo(CCMemberID memberId, CCMemberInfo* pMemberInfo);

//help functions.
    CCError GetMemberStatus(INT memberId);

//page.
    CCError IsPageRegistered(CCVisitor* pVisitorInfo);

//page and member.
    CCError CanThisMemberUseThePage(CCMemberID memberId, CCPageID pageID);
    CCError MembersCanAccessThisPage(CCPageID pageID, CCMemberID* pMemberId);
private:
    :
};

#endif/*__RESMGR_H_*/
```

A-14

```
/*-------------------------------------  ------------------------*/
/*                                                               */
/*  Stellar Server Stack Project                                 */
/*  resource management implement file                           */
/*                                                               */
/*  RESMGR.CPP                                                   */
/*                                                               */
/*  Copyright (c) 1997 Stellar Computing Corp.                   */
/*  All rights reserved                                          */
/*                                                               */
/*---------------------------------------------------------------*/
////////////////////////////////////////////////////////////////////////////
//Create Date:  06/18/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//
#include "comdefs.h"
#include "ccdefs.h"
#include "resmgr.h"

////////////////////////////////////////////////////////////////////////////
//Purpose:
//  constructor of CResourceManager
//  <TBD>
//Parameters:
//Return:
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CResourceManager::CResourceManager()
{
}

////////////////////////////////////////////////////////////////////////////
//Purpose:
//  destructor of CResourceManager
//  <TBD>
//Parameters:
//Return:
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CResourceManager::~CResourceManager()
{
}

//for accounting.
////////////////////////////////////////////////////////////////////////////
//Purpose:
//  Save time used information per login/logout into context depository.
//  will used to create report to user and determine if the member is run
//  out of time resource.
//  called when member logout and before disconnect.
//Parameters:
//  memberId. a valid member id used to save time resource usage.
//  loginTime. the login Time.
//  logoutTime. the logout Time.
//Return:
//  CC_NOERROR indicate that the time information has saved successfully.
//  CC_MEMBER_INVALIDATE_ID indicate that the member id is invalidate.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::SaveAccountingTimeInfo(CCMemberID memberId,
    CCTime loginTime,
    CCTime logoutTime)
{
    //<TBD>
    return CC_NOERROR;
}
```

A-15

```
////////////////////////////////////////.. .////////////////////////////
//Purpose:
//  Get the remain time of the member indentified by the memberId.
//Parameters:
//  memberId. a valid member id to be calculated the remain time.
//  timeRemain OUT. if successful. it will contains the remain time of this
//      member.
//Return:
//  CC_NOERROR. the call is success. remain time is in time remain.
//  CC_MEMBER_INVALIDATE_ID. the member id not right.
//  CC_MEMBER_EXPIRED. the member is a expired user. and timeRemain will set
//      to value 0.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::GetRemainTime(CCMemberID memberId. ULONG* timeRemain)
{
    //<TBD>
    return CC_NOERROR;
}

////////////////////////////////////////////////////////////////////////////
//Purpose:
//  write a accountting report to show to member.
//Parameters:
//  memberId. a valid member id to be given a accounting report.
//  fileName. a valid file name used to out put accounting report.
//Return:
//  CC_NOERROR. the call is success. report has writen to file named
//      fileName
//  CC_MEMBER_INVALIDATE_ID. the member id not right.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::WriteAccountingReport(INT memberId. LPCSTR fileName)
{
    //<TBD>
    return CC_NOERROR;
}

//member manager.
////////////////////////////////////////////////////////////////////////////
//Purpose:
//  Register a new member defined by pMemberInfo.
//Parameter:
//  pMemberInfo. a member info filled by member and checked by call center.
//      Resource manager will save this message. return a ID through the
//      memberId field.
//Return:
//  CC_NOERROR indicate that the member has register successful and all inf-
//      ormation has saved in context depository. has been assigned a user
//      ID and a password to this user.
//  CC_REGISTER_REQUIRED_FIELD_NOT_FIELD indicate some important field not
//      filled. call center will call web interface to ask member to retry.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::RegisterMember(CCMemberInfo* pMemberInfo)
{
    //<TBD>
    return CC_NOERROR;
}

////////////////////////////////////////////////////////////////////////////
//Purpose:
//  Update a exist member information. all field can be modified except me-
//  mber id.
//Parameter:
//  pMemberInfo. a member info filled by user and checked by call center.
//      Resource manager will save this message. return a ID through the u-
//      serid field.
//Return:
//  CC_NOERROR indicate that the member has register successful and all in-
//      formation has save in context depository. has assign a user ID and
```

A-16

```
//      a password to this user.
//  CC REGISTER REQUIRED FIELD NOT FIELD indicate some important field not
//      filled. call center will call web interface to show user to retry.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::UpdateMemberInfo(CCMemberInfo* pMemberInfo)
{
    //<TBD>
    return CC_NOERROR;
}

////////////////////////////////////////////////////////////////////////////
//Purpose:
//  Try to find a match member list from the context depository.
//parameter:
//  pVistor.  a vistor information.
//      pCount IN OUT. indicate a alloced memory block. If memory not enough.
//      returnCC NOT ENOUGHT_MEM. and make this to the size of memory should
//      alloc in byte.
//  pId OUT. if match successful. will contains the match result. a list of
//      member id. if not match. no affect to this parameter.
//Return:
//  CC NOERROR indicate match successful. and the result is in pId.
//  CC NOT ENOUGHT MEM indicate match successful and the memory provide is small
//      . the right size if in *pCount
//  CC MATCH FAIL indicate no match member found.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::IsThisVistorMatchAnyMember(
    CCVistor* pVistor,
    INT* pCount,
    CCMemberID* pId)
{
    //<TBD>
    return CC_NOERROR;
}

////////////////////////////////////////////////////////////////////////////
//Purpose:
//  Check the login information.
//Parameter:
//  pData IN. a login information contains user id. user password.
//Return:
//  CC NOERROR indicate that the member is a valid user. and can login.
//  CC EXPIRED USER indicate that the member is a valid user. but form some
//      reason. it has expired. one reason is that the time resource has run
//      our.
//  CC INVALIDATE MEMBER ID indicate the user id is wrong.
//  CC INVALIDATE PASSWORD indicate the password is wrong.
//      call center will let user retype this information.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::MemberLogin(CCLogin* pData)
{
    //<TBD>
    return CC_NOERROR;
}

////////////////////////////////////////////////////////////////////////////
//Purpose:
//  Get the detail member information from context depository.
//Parameter:
//  memberId. an valid member id.
//  pmemberInfo OUT. a CCMemberInfo struct alloced by caller. if this call
//      successful. will contains the user detail information.
//Return:
//  CC NOERROR indicate that the member is a valid user. and detail inform-
//      ation has copy into pMemberInfo.
//  CC INVALIDATE USERID indicate the user id is wrong. pMemberInfo keep
//      untouched.
//Create Date:  06/25/97
```

A-17

22

```
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::GetMemberDetailInfo(
    CCMemberID memberId,
    CCMemberInfo* pMemberInfo)
{
    //<TBD>
    return CC_NOERROR;
}

//help functions:
////////////////////////////////////////////////////////////////////////////
//Purpose:
//  Check if the given member is expired.
//Parameter:
//  memberId, an valid member id.
//Return:
//  CC MEMBER VALID indicate that the member is a valid user, and detail
//      information has copy into pMemberInfo.
//  CC MEMBER INVALIDATE ID indicate the member id is invalidate.
//  CC MEMBER EXPIRED indicate the member is expired.
//  CC MEMBER ACCOUNT FROZEN indicate the member's account is frozen, may
//      be required by this member.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::GetMemberStatus(INT memberId)
{
    //<TBD>
    return CC_NOERROR;
}

//page manager.
////////////////////////////////////////////////////////////////////////////
//Purpose:
//  check if the given page is registed before, page info in pVisitorInfo.
//Parameter:
//  pVisitorInfo, a visitor information, only use the page url in this call.
//  pPageId, a page id point, alloced by caller, if successful, will contains
//      the the page id.
//Return:
//  CC PAGE REGISTED indicate that the page is previou registed by tool program.
//      and the pPageId is filled by the page id of this url.
//  CC PAGE NOT REGISTERED indicate the page is not registered.
//      pPageId keep untoughed.
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::IsPageRegistered(
    CCVisitor* pVisitorInfo,
    CCPageID* pPageId)
{
    //<TBD>
    return CC_PAGE_REGISTED;
}

//page and member.
////////////////////////////////////////////////////////////////////////////
//Purpose:
//  to check if a special member can access a specail page.
//Parameter:
//  memberId, a member id used to determine whether the page can be used by the
//      member.
//  pageId, a valid page id to check if can be used by the member.
//Return:
//  CC PAGE CAN BE USED, means the page can be used by this member.
//  CC PAGE NOT REGISTERED, the page not registered.
//  CC_MEMBER_INVALIDATE_ID, the member is invalidate.
//
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
```

A-18

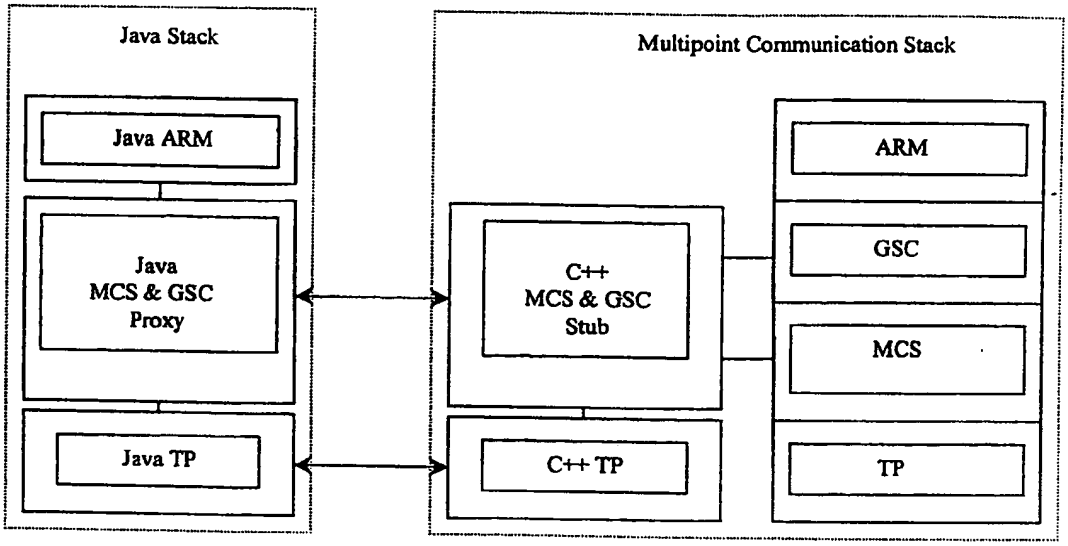
```
//      Reason:
//      Detail:
//                                                                    23
CCError CResourceManager::CanThisMemberUseThePage(
    CCMemberID memberId,
    CCPageID pageID)
{
    //<TBD>
    return CC_PAGE_CAN_BE_USED;
}

//page and member.
////////////////////////////////////////////////////////////////////////////
//Purpose:
//  To get member list that can access or want to access this page.
//Parameter:
//  pageId, a valid page id to check if can be accessed by the other member
//  pCount, a point to int, will contains the success result's member count.
//      if not enough memory, will contains the size, and return
//      CC_NOT_ENOUGHT_MEM
//  pMemberId, a member id list to contains the success result.
//Return:
//  CC_MATCH_SUCCESS, means the mathc operate successful, match result is save to
//      pMemberId.
//  CC_NOT_ENOUGHT_MEM, means the alloced memory is not enough, the right size is
//      in pCount.
//  CC_PAGE_NOT_REGISTERED, the page not registered.
//  CC_MATCH_FAILED, match failed.
//
//Create Date:  06/25/97
//Code Writer:  Tom.Wang
//Modify Record:
//      Modifier:
//      Date:
//      Reason:
//      Detail:
//
CCError CResourceManager::MembersCanAccessThisPage(
    CCPageID pageID,
    UINT* pCount,
    CCMemberID* pMemberId)
{
    //<TBD>
    return CC_NOERROR;
}
```

A-19

24

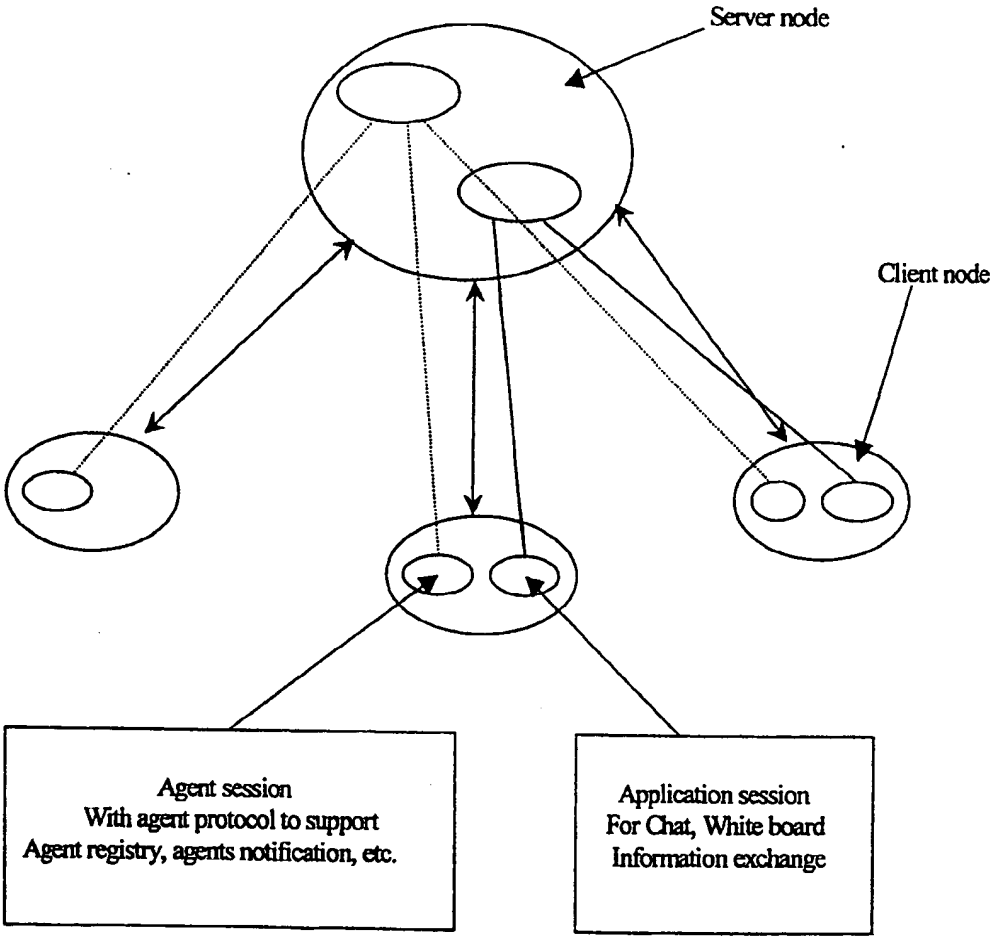
STELLAR Context Conferencing

1. Architecture



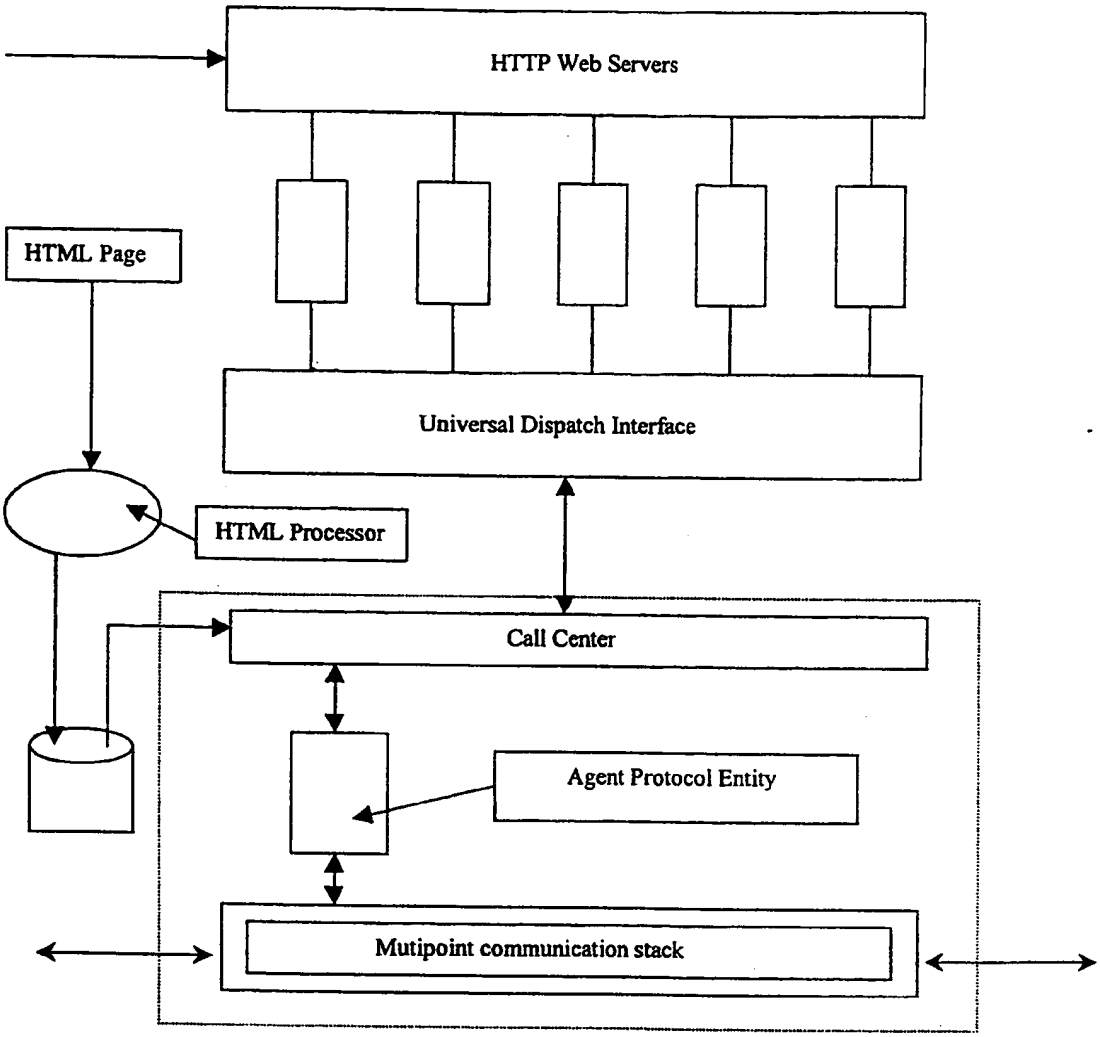
Communication Stack Architecture

TP	:	Transport
MCS	:	Multipoint Communication Service
GSC	:	General Session Control
ARM	:	Application Resource Manager



Typical running topology

A-21


HTTP Web Servers
HTML Page
Universal Dispatch Interface
HTML Processor
Call Center
Agent Protocol Entity
Mutipoint communication stack

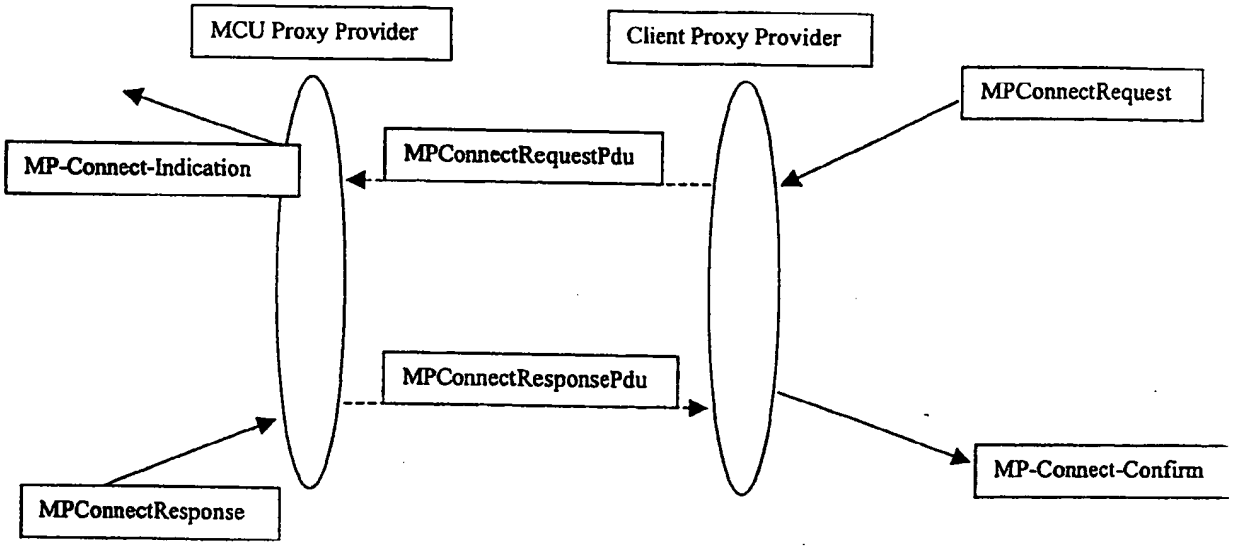
27

MCU Proxy Protocol

1. Overview

2. Connect to MCU

2.1 MPConnectRequest



On receipt of MPConnectRequest, the client proxy provider should first issue the TPConnectRequest with the address supplied in the MPConnectRequest.

On receipt of the successful TP-CONNECT-CONFIRM message, the client proxy provider should issue TPSendDataRequest with which the data field filled MPConnectRequestPdu.

Table 2.1 MPConnectRequest parameters

Name	Description
Local address	
Remote address	
MCU domain name	
Connection Handle	
User Data	

Table 2.2 MP-Connect-Request-Pdu

Contents	Source	Sink
Local address		
Remote address		
MCU domain name		
User Data		

On receipt of a TP-DATA-INDICATION message, the proxy provider should decode the message first, if it is a MPConnectRequestPdu, it should issue the MP-CONNECT-INDICATION message with the parameters filled from the data in MPConnectRequestPdu.

Table 2.3 MPConnectIndication parameters

Name	Description
Local address	
Remote address	
MCU domain name	
Connection Handle	
User Data	

2.2 MPConnectResponse

Table 2.4 MPConnectResponse parameters

Name	Description
Connection Handle	
MCU domain name	
Result	
User Data	

Table 2.5 MP-Connect-Response-Pdu

Contents	Source	Sink
MCU domain name		
Result		
User Data		

Table 2.6 MPConnectConfirm parameters

Name	Description
Connection Handle	
Result	
User Data	

2.3 MPDisconnectRequest

Table 2.7 MPDisconnectRequest parameters

Name	Description
Connection Handle	
Reason	

Table 2.8 MP-Disconnect-Indication-Pdu

Contents	Source	Sink
Reason		

Table 2.9 MPDisconnectIndication parameters

Name	Description
Connection Handle	
Reason	

3. MCS Proxy

3.1 MCSRegisterUserApplication
3.2 MCSCleanup
3.3 MCSAttachUserRequest
3.4 MCSChannelJoinRequest
3.5 MCSChannelLeaveRequest
3.6 MCSChannelConveneRequest
3.7 MCSChannelAdmitRequest
3.8 MCSChannelDisbandRequest
3.9 MCSChannelExpelRequest
3.10 MCSSendDataRequest
3.11 MCSTokenGrabRequest
3.12 MCSTokenInhibitRequest
3.13 MCSTokenPleaseRequest
3.14 MCSTokenGiveRequest
3.15 MCSTokenGiveResponse
3.16 MCSTokenReleaseRequest
3.17 MCSTokenTestRequest
3.18 MCSReadyRequest

4. GCC Proxy

4.1 GCCRegister
4.2 GCCCleanup
4.3 GCCCreateSap
4.4 GCCDeleteSap
4.5 GCCConferenceRosterInquireRequest
4.6 GCCApplicationEnrollRequest
4.7 GCCApplicationInvokeRequest
4.8 GCCApplicationRosterInquireRequest

4.9 GCCRegistryAllocateHandleRequest
4.10 GCCRegistryAssignTokenRequest
4.11 GCCRegistryDeleteEntryRequest
4.12 GCCRegistryMonitorRequest
4.13 GCCRegistryRegisterChannelRequest
4.14 GCCRegistryRetrieveEntryRequest
4.15 GCCRegistrySetParameterRequest
4.16 GCCGetConferenceLocalNodeId
4.17 GCCGetConferenceTopNodeId
4.18 GCCGetConferenceSuperNodeId
4.19 GCCGetSessionInstanceNumber
4.20 GCCGetSessionInfo
4.21 GCCGetNodeInfo
4.22 GCCReadyRequest

STELLAR Agent Framework

1. Overview

1.1 Agent

Agent is an executable object can travel over the underlying network.
Global unique naming scheme

1.2 Agent framework

Agent framework is an abstract layer. It can enable the traveling, executing of the agents, sits on each node across the entire network.

A travel itinerary for specifying complex travel patterns with multiple destinations and automatic failure handling
A white board mechanism allowing multiple agents to collaborate and share information asynchronously
An agent message-passing scheme that supports loosely coupled asynchronous as well as synchronous peer to per communication between agents
A network agent class loader that allows an agent's Java byte code and state information to travel across the network, an execution context that provides agents with a uniform environment independent of the actual computer system on which they are executing.

1.3 Agent provider

Agent provider sits on each node, it participates the agent community, provides the running environment of the arriving or local agents.

Provides multiple services
    Services can be retrieved from a server
Agent daemon
    Running in every machine an agent will travel to.
Local native interface:
    Local execution of a program(.exe)
    Read/write local files
    Retrieve/update/delete/add data form /to database.
    Script language support
    Install/uninstall OCX/COM objects for Windows

1.4 Agent provider protocol

Agent provider protocol is the language for which the agent providers use to talk.

1.5 Agent provider session

Agent provider session is a runtime provider sociality which is hosted by the top agent provider.

1.6 Agent security

Agent proxy : a placeholder for an agent to control access to the agent

1.7 Agent repository

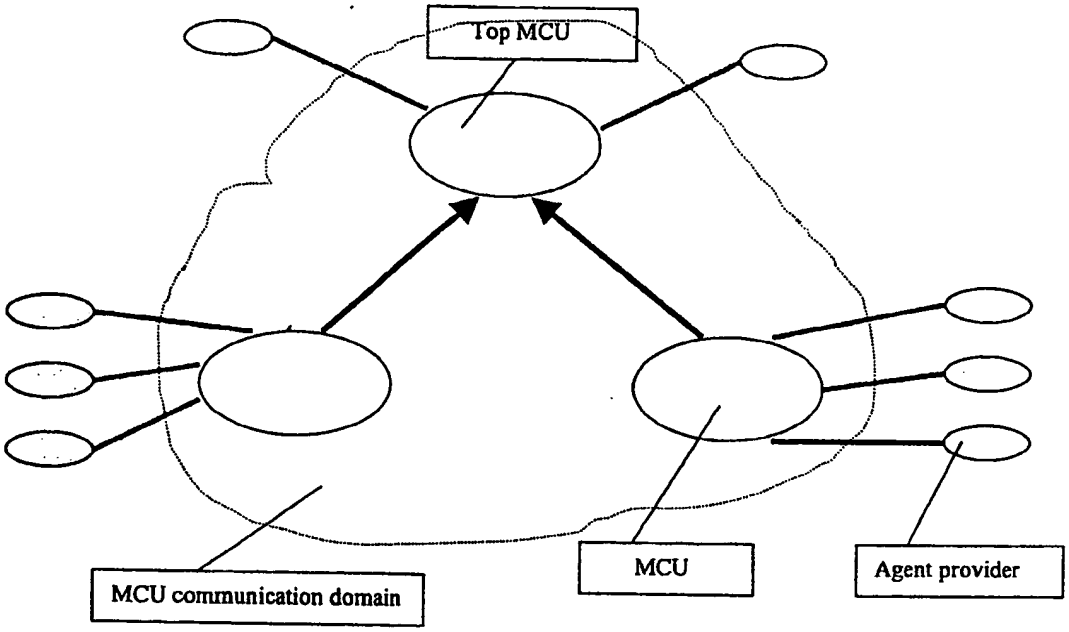
Place to save and manage all agents

1.8 MCU

MCU, Multi-point Communication Unit supplies a real-time multi-point connection service.

2. Agent framework Protocol

2.1 Setup agent community



2.1.1 MCU communication domain

MCU communication domain is the backbone of the agent sociality, its creation is beyond of this document.

2.1.2 Bind to MCU communication domain

An agent provider should first bind to the MCU communication domain, then can it start its agent service.

2.1.2.1 BindRequest

On receipt of BindRequest primitive, the agent provider should issues a MPConnectRequest to the MCU with the user data field of MPConnectRequest filled with BindRequestPdu.

Table 2.1 Bind-Request-Pdu

Contents	Source	Sink
Agent provider name	Request(M)	Indication(M)
Agent provider password	Request(O)	Indication(C)
User Data	Request(O)	Indication(C)

On receipt of MPConnectIndication, the agent provider should decode the BindRequestPdu in the user data field and issues BindIndication primitive with following parameters

Table 2.2 Bind-Indication parameters

Name	Description
Agent provider name	Agent name which issues the connect request, this name is locally unique in The indication node
Agent provider password	The logon password for the provided name
Bind handle	Handle for identifying the bound connection

2.1.2.2 BindResponse

Table 2.3 BindResponse parameters

Name	Description
Bind handle	Handle for identifying the bound connection
Result	Specifies whether the request is successful

On receipt of BindResponse, the agent provider should issues a MPConnectResponse primitive with the user data field filled with BindResponsePdu

Table 2.4 Bind-Response-Pdu

Contents	Source	Sink
Result	Response(M)	Confirm(M)
User Data	Response(O)	Confirm(C)

Table 2.5 BindConfirm parameters

Name	Description
Bind handle	Handle for the bound connection
Result	Specifies whether the request is successful

2.1.2.3 UnbindRequest

Table 2.6 UnbindRequest parameters

Name	Description
Bind handle	Handle for the bound connection
Reason	

Table 2.7 Bind-Response-Pdu

Contents	Source	Sink
Reason		

2.2 Agent provider session

2.3 Top Agent provider

2.4 Register Agent

2.5 Agent Provider primitives

3. Agent PDU definition

4. Agent provider API (COM)

5. Agent mobile API (COM)

Agent Server Design

Agent profile
- a globally unique naming scheme for agents

Agent server wire protocol
- Currently, it uses T.120 ARM interface, but can be replaced with another wire protocol(e.g. TCOP/IP, HTTP or IPX/SPX).

Agent communication protocol
- GetAgent
- DispatchAgent
- RetractAgent
- PutAgentData
- QueryAgentStatus

Agent local service provider:
- Provide multiple services.
  - Services can be retrieved from a server
- Agent daemon
  - Running in every machine an agent will travel to.
- Agent local services:
  - Agent execution failure handling and recovery(Agent persistence support)
  - Local native interface:
    - local execution of a program(.exe)
    - read/write local files
    - Retrieve/update/delete/add data from/to database.
    - script language support
    - install/uninstall OCX/COM objects for Windows

Agent application protocol
- Call Center Application Protocol

Agent security
- Agent proxy: a placeholder for an agent to control access to the agent

Agent infrastructure
- a travel itinerary for specifying complex travel patterns with multiple destinations and automatic failure handling,
- a white board mechanism allowing multiple agents to collaborate and share information asynchronously,
- an agent message-passing scheme that supports loosely coupled asynchronous as well as synchronous peer-to-peer communication between agents,
- a network agent class loader that allows an agent's Java byte code and state information to travel across the network,
- an execution context that provides agents with a uniform environment independent of the actual computer system on which they are executing.

Agent Author:
- Agent authoring tool

Agent Repository:
- Place to save and manage all agents

Agent script language

Agent service provider for Windows platform:

The agent service provider in Windows platform is a COM server which supports multiple agent COM interface aggregation and agent retrieval from an agent server. An agent service user(either an agent or a local application with a COM interface) can query agent services from the provider. The provider will query all the local registered agents to get all of their interfaces and aggregated through it. An agent service in Windows platform is a COM interface.

When an agent user queries a specific agent service(e.g. read from a specific database), the provider will query all the local registered agent services and find out if there is one with that service. If there is one available, it will provider to the user that interface and manages reference counts for interfaces. If there in no service(COM interface) available, it will provide a callback , then go ahead to request that server from agent servers. If the service is available from a server, the agent is downloaded and it calls back the user for that service. Otherwise, it notifies the user that the service is not available.

All the agents downloaded from servers are registered locally for all the interfaces available. The provider controls memory/disk usage, too. If the usage exceeds the limit, it will automatically delete least used agents until the usage is below the limit. All the services for deleted agents are unregistered. When a user requests a service not locally available, it will request from an agent server. When it gets the service, it downloads that service, saves it locally and registers its services.

Agent communication protocol
- GetAgentRequest: a provider to get an agent service from a server
  - Indication
  - Response
  - Confirm
- PutAgentDataRequest: a provider to put data back from an agent execution
  - Indication
  - Confirm
- DispatchAgentRequest: a server to dispatch an agent to a provider
  - Indication
  - Confirm
- RetractAgentRequest: a server to retract an agent from a provider
  - Indication
  - Confirm
- QueryAgentStatusRequest: a server to query for the status of an agent
  - Indication
  - Response
  - Confirm

Scenarios:

QD: Create an agent with AgentAuthor
- Specify agent activity
  - COM interface to a QD program(e.g. Fix-It)
  - Gather information/system data
    - (e.g. memory, disk space, hardware/software, registry)
  - Run a program
  - Run a script block
- Specify an itinerary

Dispatch the agent

Report result and update database

S3: Genie(animation, speech recognition, text to speech) support

Call Center Module Interface Design

Listener to Call Center Interface:
- Listener Management
  - GetListenerList
  - GetDetailListenerInfo
  - RegisterListener
  - UnregisterListener
- Session Management
  - GetSessionList
  - GetDetailSessionInfo
  - JoinSession
  - LeaveSession
  - TransferSession
  - InviteToSession
- Resource management
  - GetResourceList
  - GetDetailResourceInfo

Call Center To Listener Interface:
- Session Management
  - StartSession
  - InvitedToSession
  - TransferredToSession
  - FinishSession

Resource To Call Center Interface:
- resource management
  - RegisterResource
  - UnregisterResource
  - GetResourceList
  - GetDetailResourceInfo

Guest To Call Center Interface:
- Session Management
  - ActivateResource
  - JoinSession
  - LeaveSession

Call Center To Guest Interface:
- Session Management
  - StartSession
  - FinishSession

Call Center To Database Interface:
- SaveResource
- UpdateResource
- DeleteResource
- LoadResource
- SaveListenerList
- LoadListenerList

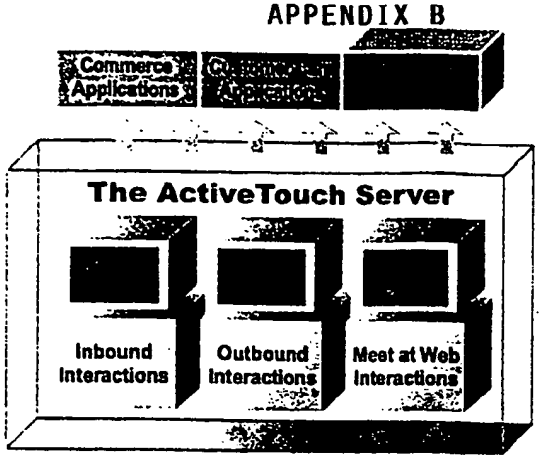
Call Center To Agent Server Interface:
- RegisterAgentSap
- UnregisterAgentSap
- RegisterAgent
- UnregisterAgent
- SendAgentMessage

Agent Server To Call Center Interface:
- ReceiveAgentMessage

Objects:

- AgentServerInterface
- DatabaseMgr
- CallCenterApplicationProtocol
  - GetListenerList
  - GetDetailListenerInfo
  - RegisterListener
  - UnregisterListener
  - GetSessionList
  - GetDetailSessionInfo
  - JoinSession
  - LeaveSession
  - TransferSession
  - InviteToSession
  - GetResourceList
  - GetDetailResourceInfo
  - StartSession
  - FinishSession
- CallCenterMgr
  - AgentServerInterface
  - DatabaseMgr
  - ResourceMgr
  - SessionList
  - ListenerList
  - CallCenterApplicationProtocol
- ResourceMgr
  - ResourceList
- Session
  - ParticipantList
- Resource
- Participant
- Listener inherited from Participant
- Guest inherited from Participant

APPENDIX B



Architectural Strength

- Scalability and Customization
- Tight Integration of Data and Telephony
- Full Web Integration via HTTP tunneling
- Comprehensive Security
- Open Standards (HTTP, TSAPI, LDAP, T.120, ODBC, Encryption)
- High Performance
- Fault Tolerant

"The Three Cs" of Collaborative Computing

The ActiveTouch Server enables Web solutions developers to deliver a new class of enterprise computing: Web-based shared workspaces that dramatically improve communication effectiveness and add value to the bottom line. Our collaboration application server offers robust functionality for all the critical enterprise interactions: inbound, outbound, and meet-at-Web. With a feature-rich toolkit offering collaborative application assembly, the server enables the 3 C's of Collaborative Computing Applications: Commerce, Customer Care and Conferencing applications.

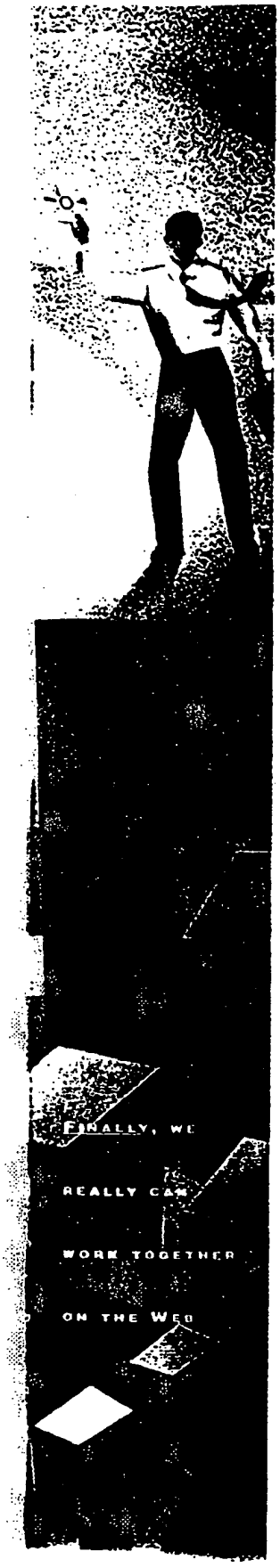
The ActiveTouch Server and its application toolkit redefines the meaning of virtual interactivity: A click of a button brings all parties together via phone and browser. Within the ActiveTouch system, users can instantly and securely access corporate databases and intranets, run software demos, review billing information, resolve technical issues, or provide real-time customer service and support. Conferencing users can jointly view the same pages on a site, review and annotate the same documents, concurrently plan group-project tasks, and save and print collaboratively changed documents.

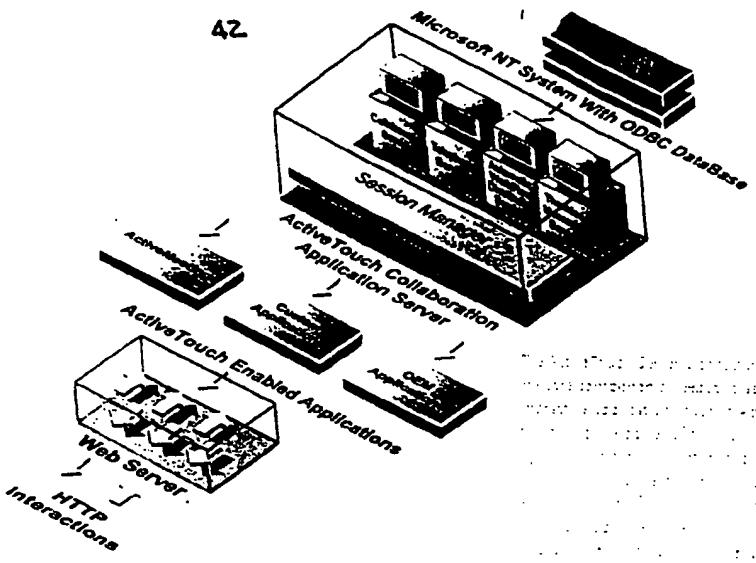
ActiveTouch Server Exposed

ActiveTouch Server applications are browser-independent, and require no manual client-side installation. Their easy deployment across an enterprise network (and smooth integration with third-party front office and call-center applications) belies the server's power. The server is comprised of five core components: Distributed Data Collaboration Services, Distributed Telephony Services, Automatic Workflow Distribution Services, Threaded Messaging Services, and Session Manager. In addition, the Web Client Services supports the browser-independent components used for building client applications.

These five core components can be assembled independently or collectively to create customized collaborative applications. Each server component can be distributed across separate machines or all run on the same machine. The core components have considerable strengths:

Collaboration Services provide the core data services for sharing and annotating text, images, applications and Web pages using advanced HTTP tunneling. Based on a T.120 protocol-compatible stack, the Collaboration Server supports the ActiveTouch browser client as well as Microsoft NetMeeting. It also includes the MultiServer coupling system that provides tight clustering for scalability and fault tolerance.





Distributed Telephony Services enable complete browser-controlled telephony functions such as teleconferences, invite, forward, and meet-me service, in addition to PBX integration. The telephony server can be remotely located from the rest of the ActiveTouch Server to allow for telephone connections to be placed from appropriate remote locations.

Automatic Workflow Distribution Services provide intelligent call and data routing for sales and support applications and has an API for easy integration with existing call center applications. The server provides Automated Call Distribution (ACD) functionality for Web-initiated calls.

Threaded Message Services allow website visitors to drop off messages in lieu of live voice interactions. For example, website visitors can drop off messages for particular consultants (or a category of consultants) on websites.

Session Manager provides conference setup, resource allocation, and multi-tiered security and encryption. Enterprise integration API's allow data sessions to be tightly integrated with enterprise business objects (e.g., PO's, change orders) and directories as well as providing direct access to databases through an ODBC interface.

Other key features include:

Web Client Manager supports the browser-independent toolkit for building collaborative applications.

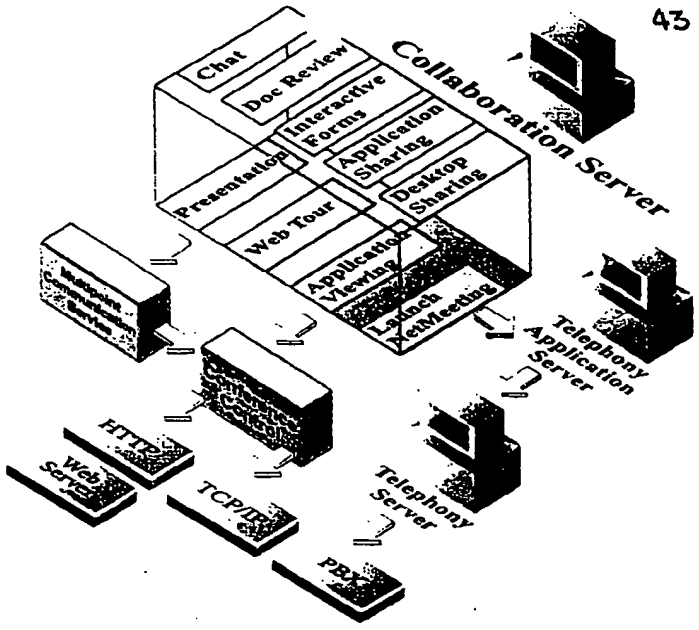
Administration Module allows real-time monitoring of live sessions and account management, useful for customer care services.

Optional Billing Module tracks real-time connections and drop-off messages for billing purposes and provides the ability to integrate with existing billing systems.

ACTIVETOUCH SERVER FOUNDATION

- Live Data Collaboration Application Server for Real-Time Commerce, Customer Care and Conferencing
- Industrial-Strength CTI and T.120 Services over HTTP
- Built for Enterprise and Service Class Applications
- Secure, Scalable, Robust and Reliable
- Browser Ease of Use and Familiarity – 100% Web-based

B-2



THE ACTIVETOUCH SERVER COLLABORATIVE TOOLKIT

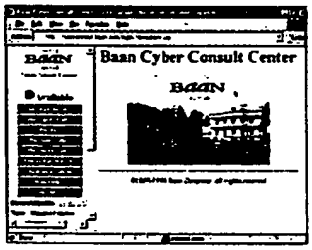
- Schedule and Manage Web-based Meetings
- Share, Edit, Annotate and Save Documents
- Share Presentations and Collectively Tour the Web
- Share Applications and Software Demonstrations
- Desktop Sharing and Remote Control
- Share Whiteboards
- Complete Interactive Forms
- Swift and Complete Screen Synchronizations for all Data Transmissions
- Launch Microsoft NetMeeting as a server session
- Billing Option for Phone Sales and Support
- Complete Integration with Corporate DBMS's via ODBC

ActiveTouch Application Scenarios

The ActiveTouch Server's robust set of network services and the power of its synchronous and asynchronous information exchange can bring significant and cost-effective benefits for all kinds of enterprise activities. The case study and scenarios below illustrate how the server can be used to leverage your existing Web-based infrastructure and holistically manage enterprise resources.

ActiveTouch Connects Baan Consultants to Support All Sides of the Globe

Baan, one of the world's leading providers of scalable Enterprise Resource Planning (ERP) software solutions, adopted the ActiveTouch Server for use in the company's Cyber Consult service. The Cyber Consult strategy created an Internet gateway for customers to electronically access the expertise of Baan consultants, via the ActiveTouch Server. ActiveTouch allows customer project managers to instantly connect with highly qualified Baan Cyber Consult professionals available on the Web and by phone.



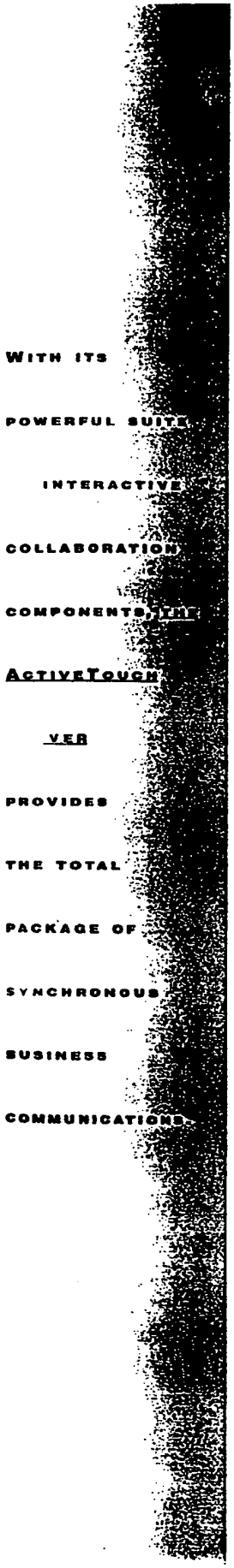
ActiveTouch Server features such as Desktop Sharing allow Baan Cyber Consult professionals to configure ERP systems directly over the Web, and review project flow charts and milestones together with customers from anywhere in the world. Cyber Consult is helping Baan differentiate itself in the highly competitive enterprise software market by reducing the travel time and cost for its consultants, while increasing the amount of time they spend working with customers.

Some Typical Applications of the ActiveTouch Server

The scenarios below illustrate a small portion of the server's power. The ActiveTouch Server can be easily customized for all of the applications below and for many more. It's a true platform for the widest range of Web-based interactive applications.

Streamline Sales: A salesperson uses an ActiveTouch application to give a customized presentation to a prospective customer over the Web. The prospect asks if his technical person at another location can join the meeting, and he is quickly connected.

As the salesperson demonstrates the product, the prospect and technician ask questions. To answer the technician's ques-



AA

THE ACTIVETOUCH WEB COLLABORATION ENVIRONMENT

- Completely browser-based for users and administrators
- Requires no manual client installation or complex user configuration
- Very easy to learn and use
- Compatible with most firewalls and proxy servers
- Seamless and integrated PSTN compatibility for reliable and familiar audio service
- Robust, reliable and scalable server-centric; services data collaboration
- Easy integration with call centers and range of enterprise applications

tions, she calls up a data sheet outlining the technical specifications. The salesperson then directs a Web-tour of the competitor's Website to demonstrate that their product is not comparable.

Asked about the price of the product, the sales person opens a spreadsheet outlining pricing options. When the prospect decides to purchase the product, the salesperson helps him complete a form on the Web to complete the sale.

Improve Channel Management: A global enterprise sales force travels frequently and works in disparate locations. Communication often required multiple phone calls and e-mail messages to different people – a situation that seldom provided satisfactory support and did not allow management to measure how distributors were being served.

With ActiveMeetings – a customized conferencing application built on the ActiveTouch Server – the business unit initiated regular distributor conferences to keep them up-to-date on new developments. Distributors now go to a single location on the Internet and check who is available in a particular department, and then connect to that person. If the person is unavailable, the distributor can leave a message and anyone in that group will be able to follow-up.

Enhance Customer Support: A customer has a software problem. Instead of explaining a solution or trading messages by e-mail or over a chat line, the specialist uses the ActiveTouch Server to view the customer's screen and identify the trouble. Using the ActiveTouch desktop-sharing capability, the specialist takes control of the customer system and fixes the problem on the spot.

He refers the customer to an electronic copy of the manual and points out relevant information that could circumvent the same problem occurring in the future. If the specialist comes across a problem outside of his area of expertise, he draws upon the expertise of a colleague in another location using the server's one-to-many conference-call feature.

Maximize Consulting Services: Running against a tight deadline, a designer is able to get feedback from several reviewers working in different parts of the country. When the concepts are ready, he uses the ActiveTouch Server to see if the project lead is available, without worrying about where the lead is working at the moment. If available, the ActiveTouch Server connects them. The designer shows the designs to the lead online, who indicates what she would like changed. When the suggestions are incorporated, the lead invites the manager to see the designs. Minor changes are made and the final design is approved.

Those are just a few of the capabilities provided by the ActiveTouch Server. All kinds of potential data-collaboration applications are possible, such as supply chain management, corporate training, telemedicine, distance learning, and countless others. Truly, sharing information on an internal website or the Internet is as easy as clicking a button – but the enterprise implications are profound.

Technical Information

T.120 Protocol Capabilities

The ActiveTouch server exploits the capabilities of T.120, a suite of networking protocols established by the International Telecommunications Union between 1993 and 1995 for multipoint data communications, multicasting, and application sharing. T.120 is rapidly gaining support on the Web as a set of industry standards for implementing real-time multimedia conferencing environments.

The T.120 data-conferencing services in the ActiveTouch Server support both TCP/IP clients such as NetMeeting and the HTTP-based ActiveTouch client.

Security

With a public key infrastructure as a foundation, the ActiveTouch Server delivers consistent security across the server and its components developed for Web servers and browsers. The ActiveTouch Server offers unsurpassed access-control granularity for data collaboration, from initial access into a server for hosts, joining a meeting, and unlisting and locking a meeting, to protecting individual documents and applications being shared.

Security Features:

- User password protection
- Session password and predefined key-entry
- Unlisted sessions
- Session lock and unlock functions
- Secure Socket Layer (SSL) supported server
- Public/private key-based encryption

Web Integration

- Browser access to all functionality through HTML/Plug-in interfaces
- Consistent functionality and UI between Netscape and Microsoft Explorer browsers
- Comprehensive HTTP protocol support
- Firewall and proxy-server friendly
- Customizable UI through script-driven interface

Performance

- Capable of supporting tens of thousands of simultaneous voice and data connections
- Collaboration data compression
- NT kernel-level integration
- Vector-based shared documents
- All servers can run on separate machines

Enterprise Software Compatible

- Architected for Enterprise Business Process Integration
- Business object-based routing
- Business object sharing within sessions
- Archiving and retrieval of sessions into enterprise software applications

Advanced Telephony and Workflow Features:

- Customizable rules-based connection-routing, call-back, and call forwarding
- Connection waiting and conferencing connections
- Multi-user/multi-session with many-to-many connections, and dynamic join and follow
- Manages the availability of call center associates
- TAPI/TSAPI interface to PBX
- Supports PSTN/ISDN/T1 interfaces

B-5

Browser Components

The ActiveTouch Server delivers its full communications functionality to conferencing clients/attendees by downloading small ActiveX controls or Netscape plug-ins to the client machine, dependent on configuration. Many ActiveTouch features are also capable of being delivered via pure html interfaces. The server performs any necessary dynamic versioning control (including all driver support) quickly and transparently for subsequent sessions. HTTP tunneling provides compatibility between the browser components and most firewall and proxy-server configurations, making the ActiveTouch Server the most universally accessible collaboration solution on the market.

ActiveTouch Installation and Support

The ActiveTouch Server can be installed at your corporate site or hosted as an outsourced service by ActiveTouch. We will provide prepackaged embeddable logic for Web pages, which allows easy incorporation of ActiveTouch functionality into Web applications by other solutions providers and enterprise customers.

We also provide templates for different applications, allowing you to build customized versions through incremental changes. Templates for customer support, sales, supplier chain, demand chain and consulting applications are available, as well as server maintenance contracts. For information and pricing on all ActiveTouch products and services, please use the contact information on this page.

About ActiveTouch

ActiveTouch recognizes that today's information exchange must have a global reach. Cost-effective, reliable methods to transact commerce, conduct meetings and share information through a combination of Web, telephone and network technology will be ever more critical for business success. To that end, we have developed tools to transform the Web into a live, interactive environment for collaboration and commerce, for businesses of all kinds.

ActiveTouch sells its platform and solutions directly to corporations and organizations, as well as OEM's. ActiveTouch solutions enable our customers to stay competitive, provide better customer service, and attract and retain profitable customer relationships.

To obtain information about licensing the ActiveTouch Server and using it to develop solutions, as well as other ActiveTouch products and services, please call us at (408) 732-8046 or visit our website at www.activetouch.com. We look forward to discussing the future of communications with you.

ActiveTouch, Inc.
1270 Oakmead Parkway, Ste. 301
Sunnyvale, CA 94086
Tel: (408) 732-8046
Fax: (408) 732-2048
E-mail: info@activetouch.com
http://www.activetouch.com/

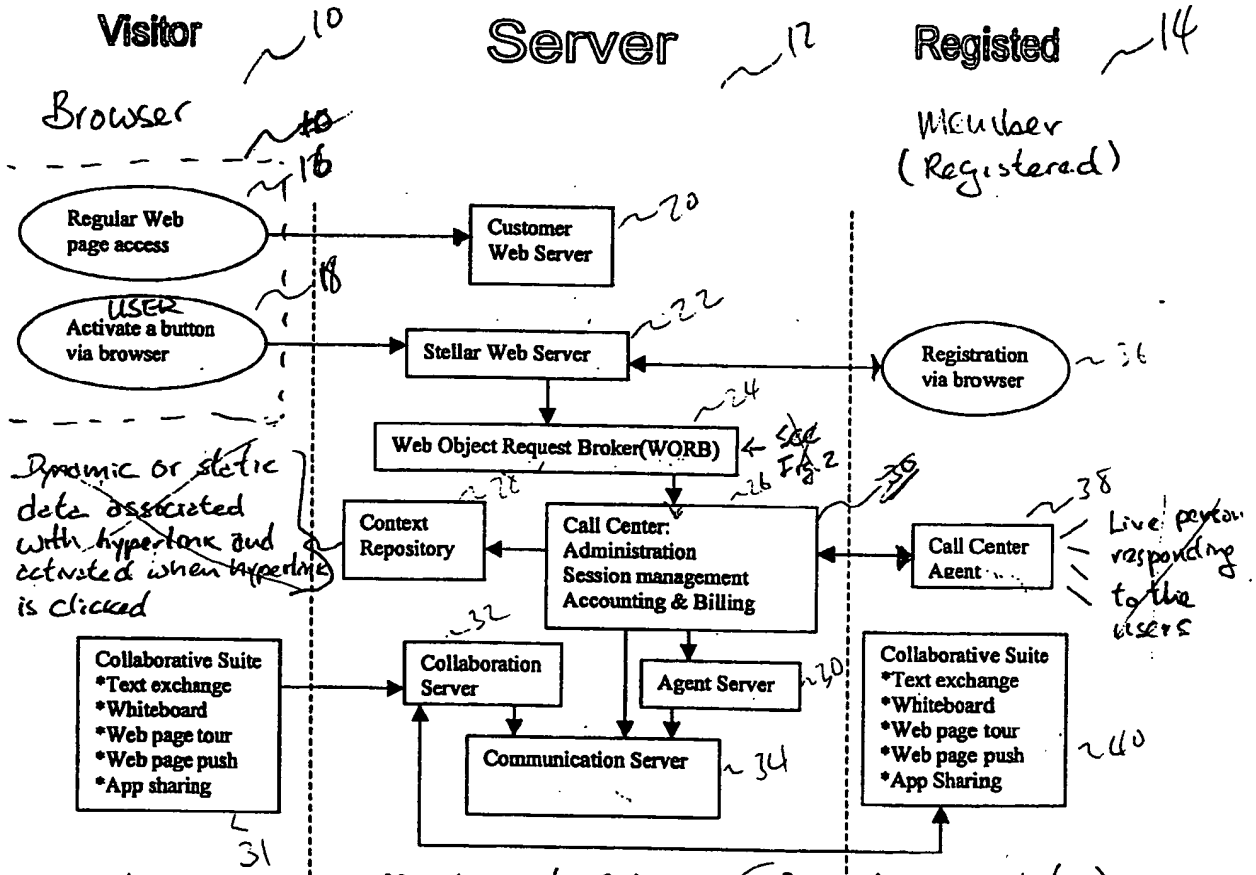
46

CLAIMS

1. A method for establishing a call center over the internet, comprising the steps of:
   a) providing a clickable link on a web page serving as a request;
   b) routing said request over the internet to a server;
   c) determining an available terminal for serving said request; and
   d) linking said request with said available terminal.

1/8

Stellar Server



Agent server - manage different agents (check configuration, virus, etc.) generally software for doing various tasks regularly.

Communication Server - connect multiple people.

Call center agent - software to allow a person to communicate to the user via phone, text exchange or other means (as listed in collaborative suite)

Fig. 1

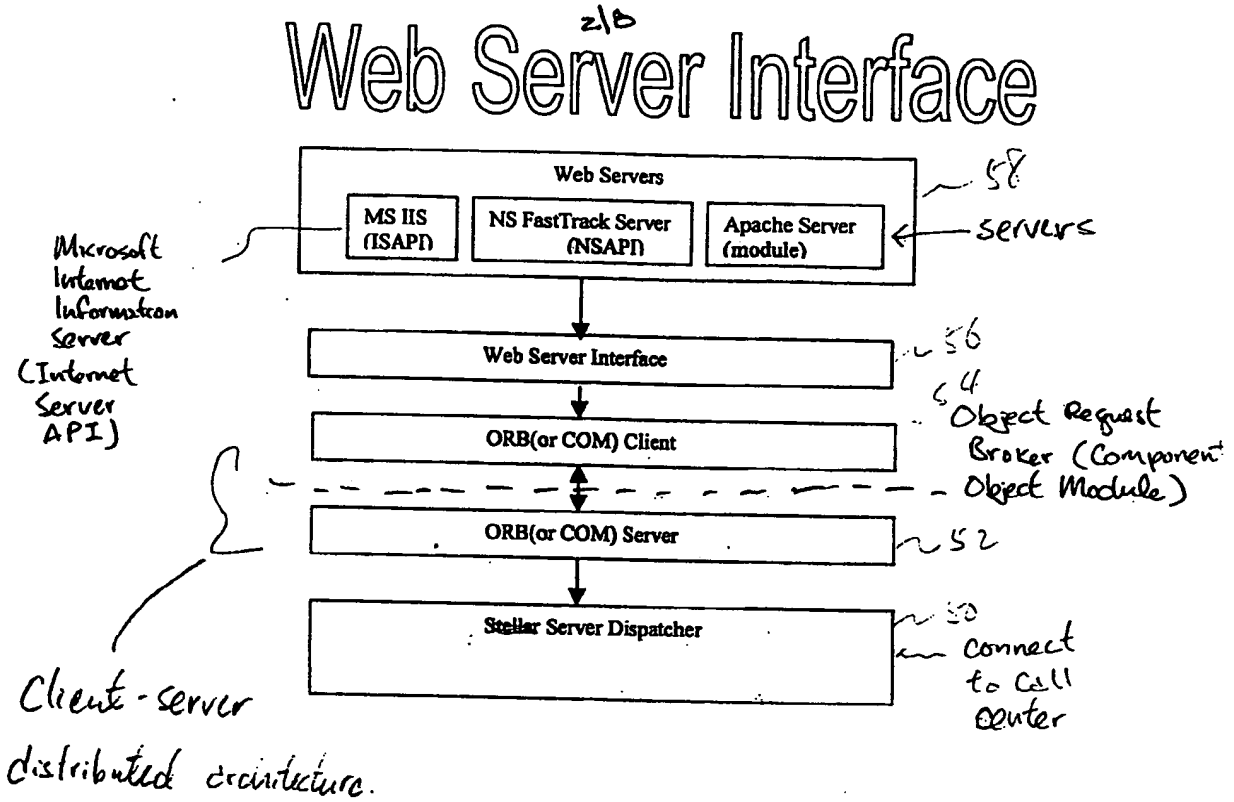

Web Server Interface
Web Servers
MS IIS (ISAPI)
NS FastTrack Server (NSAPI)
Apache Server (module)
servers
Microsoft Internet Information Server (Internet Server API)
Web Server Interface
ORB(or COM) Client
Object Request Broker (Component Object Module)
ORB(or COM) Server
Stellar Server Dispatcher
connect to Call Center
Client - server distributed architecture.

Fig. 2

Web Server Interface

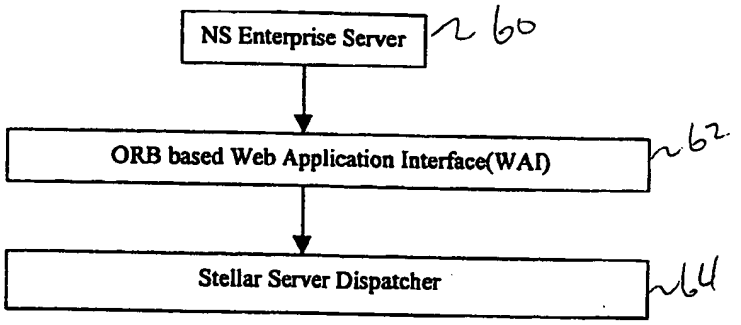


Fig. 3a

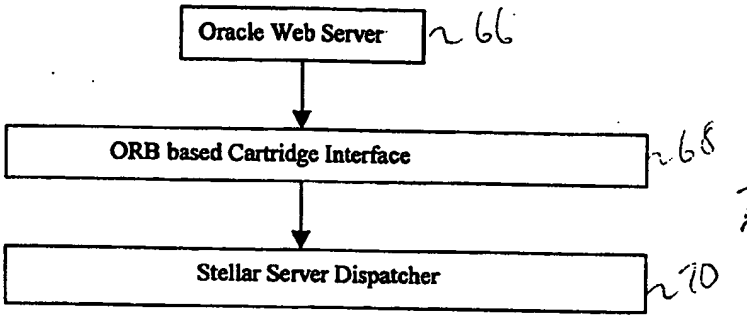
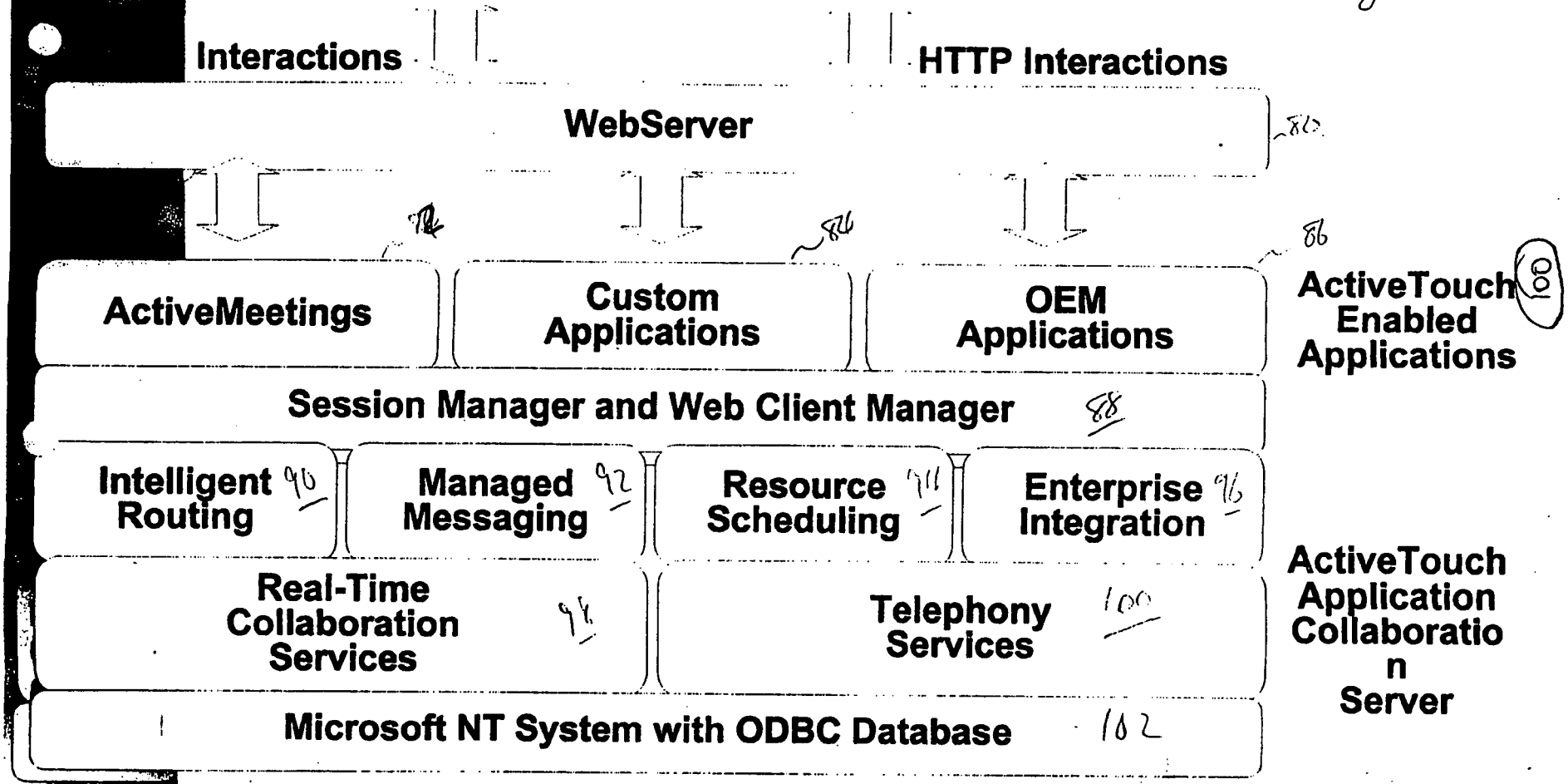


Fig. 3b

ActiveTouch Server Architecture

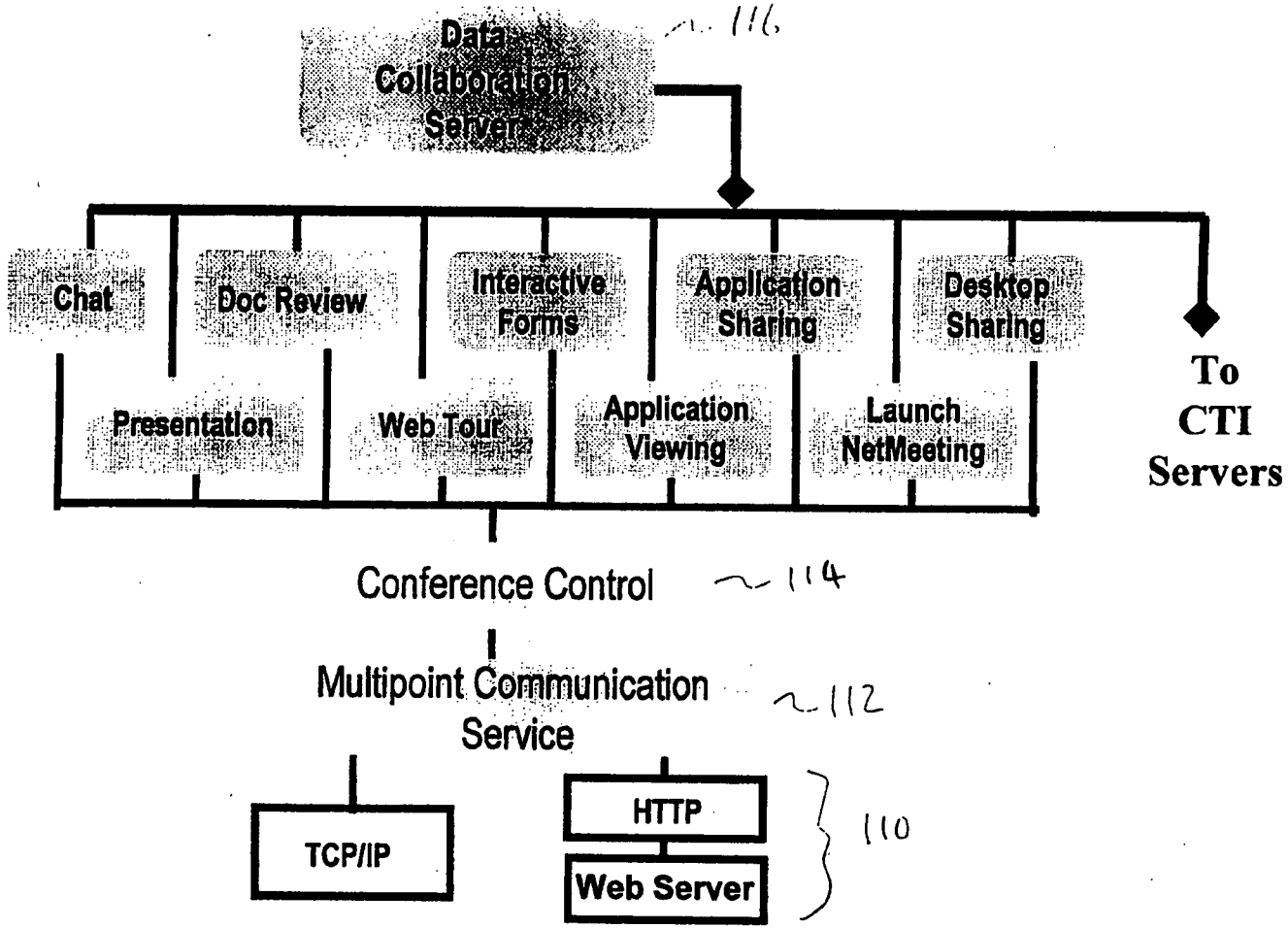
Fig. 4



ACTIVE TOUCH

Data Collaboration Services

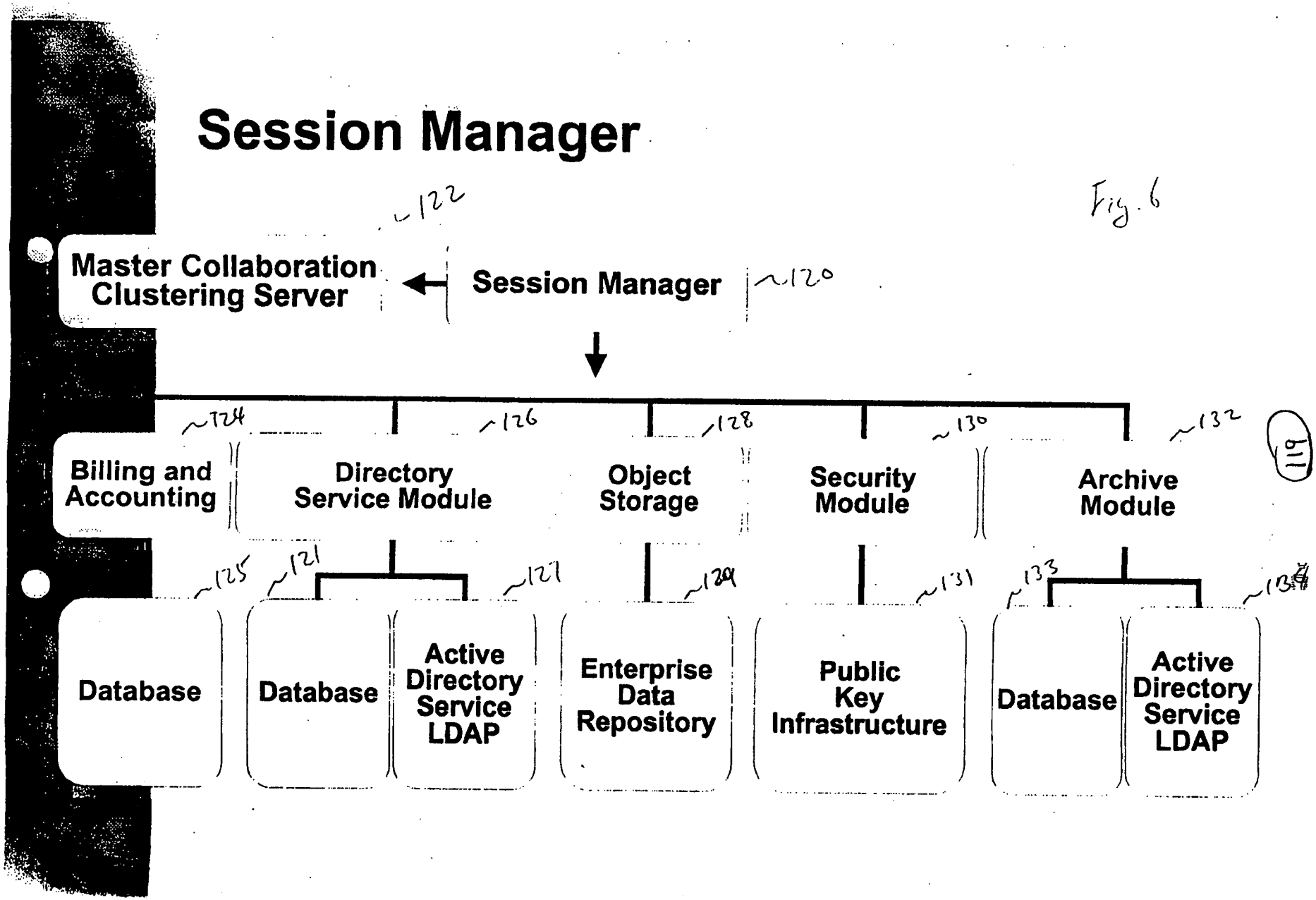
Fig. 5



5/8

811

Fig. 6


Session Manager
122
Master Collaboration Clustering Server
Session Manager
120
124
126
128
130
132
Billing and Accounting
Directory Service Module
Object Storage
Security Module
Archive Module
119
125
121
127
129
131
133
135
Database
Database
Active Directory Service LDAP
Enterprise Data Repository
Public Key Infrastructure
Database
Active Directory Service LDAP

Intelligent Routing Server

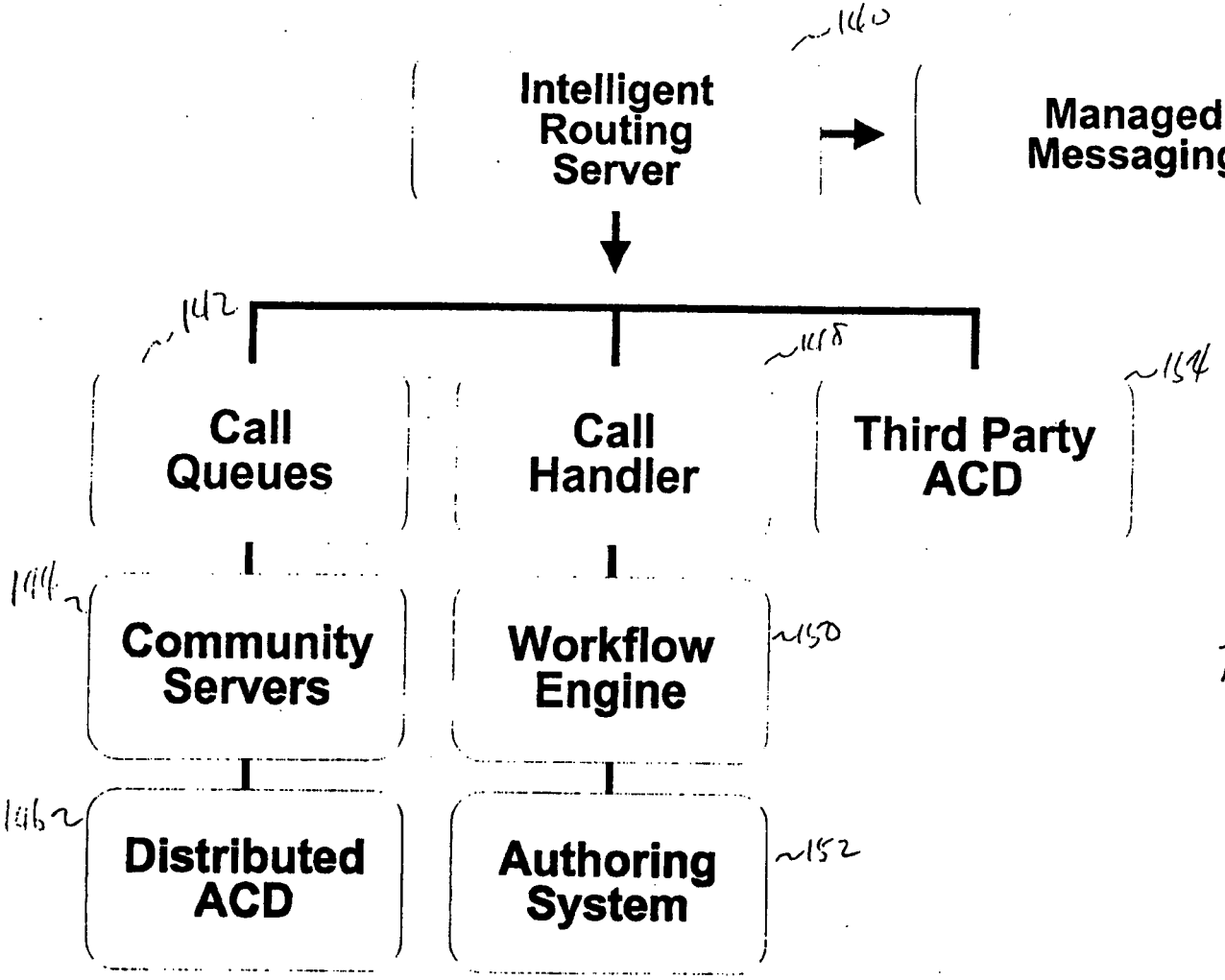


Fig. 7

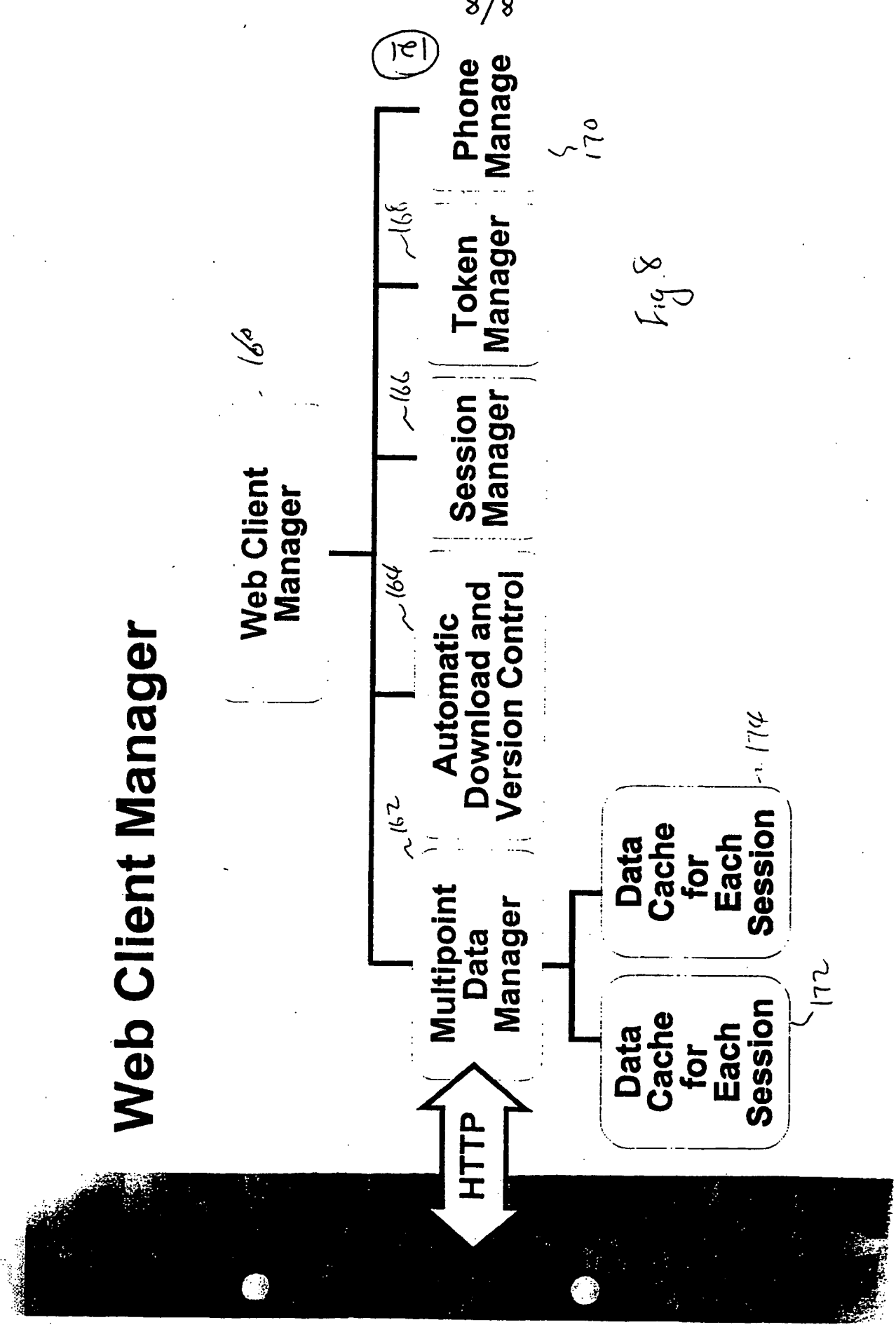

Web Client Manager
Web Client Manager
Multipoint Data Manager
Automatic Download and Version Control
Session Manager
Token Manager
Phone Manage
HTTP
Data Cache for Each Session
Data Cache for Each Session

Fig. 8